IMPLEMENTASI KOMPUTASI PARALEL DALAM PERGERAKAN LENGAN ROBOT MENGGUNAKAN INVERSE KINEMATIC

SKRIPSI
KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Tezza Rangga Putra
NIM: 135150300111015

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

JUDUL SKRIPSI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer
Keminatan : Teknik Komputer

Disusun Oleh :
Tezza Rangga Putra
NIM: 135150300111015

Skripsi ini telah diuji dan dinyatakan lulus pada
10 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Rizal Maulana , S.T., M.T., M.Sc.	Wijaya Kurniawan, S.T, M.T
NIK: 2016078910091001	NIK: 19820125 201504 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Januari 2018

Tezza Rangga Putra

NIM: 135150300111015

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan atas rahmat dan karunia Tuhan yang Maha Esa, sehingga penulis dapat menyelesaikan skripsinya dengan judul "Implementasi Komputasi Parallel dalam Pergerakan Lengan Robot menggunakan *Inverse Kinematic*". Pelaksanaan penelitian ini bertujuan untuk memenuhi salah satu syarat memperolehgelar sarjana komputer di Universitas Brawijaya serta dengan adanya penelitian ini diharapkan dapat bermanfaat bagi masyarakat. Terselesaikannya pelaksanaan penelitian ini tentu tidak lepas dari dukungan berbagai pihak, oleh karena itu penulis ingin mengucapkan terimakasih kepada

1. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku ketua jurusan Teknik Informatika.
2. Bapak Rizal Maulana, S.T., M.T., M.Sc dan Bapak Wijaya Kurniawan, S.T, M.T selaku dosen pembimbing pertama dan dosen pembimbing kedua yang terus memberikan saran, petunjuk dan bimbinga dalam proses pelaksanaan penelitian ini.
3. Seluruh staff serta dosen Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberikan fasilitas, pelayanan serta ilmu yang sangat bermanfaat bagi penulis.
4. Nanik Rusmiati dan almarhum Suyono, orangtua yang selalu mendukung serta menemani dalam berbagai usaha yang penulis lakukan hingga terselesaikannya tugas akhir ini.
5. Sofi Hanifah yang selalu memberikan dukungan semangat dan motivasi untuk mengerjakan penelitian ini hingga akhir.
6. Komunitas Robotika yang mewadahi dan memfasilitasi penulis dalam melakukan penelitian.
7. Teman – teman 5manusia, ampelgading lab, lontong (mas galih, mas syafik, mas andrika, imam, mbeek, oki, anung, najib, kun, kukuh, anin, haekal, dito, bibur, bebeks dll) yang selalu mendukung moral dan material dalam pengerjaan penelitian tugas akhir ini.

Malang, 10 Januari 2018

Penulis

tezza.rp27@gmail.com

ABSTRAK

Metode gerak lengan robot secara konvensional adalah dengan kinematika maju (*forward kinematic*), dimana dilakukan pergerakan dengan menentukan simpangan waktu masing masing sendinya agar mencapai suatu titik. Berbeda dengan kinematika balik (*inverse kinematic*), dimana pergerakan yang dilakukan dengan menentukan besar sudut yang diperlukan untuk melakukan suatu gerak. Kinematika maju sering dianggap metode yang memanfaatkan *trial and error* dimana metode ini kurang efektif karena memakan waktu yang lama. Sementara itu *inverse kinematic* menawarkan solusi yang lebih praktis dengan perhitungan komputasi yang lebih kompleks. Dengan semakin berkembangnya mikrokontroler kompleksitas komputasi pada *inverse kinematic* dapat di kerjakan dengan lebih cepat dengan menggunakan komputasi parallel yang dapat memanfaatkan seluruh *core* yang ada pada mikrokontroler. OpenMP adalah direktori program C++ yang menggambarkan penggunaan antarmuka program aplikasi untuk melakukan perhitungan paralel di lingkungan memori secara bersamaan. OpenMP menjadi salah satu program yang dapat dimanfaatkan untuk melakukan komputasi parallel. Hasilnya OpenMP mampu membagi program sehingga Program sekuensial membebankan prosesnya pada core 1 (30,74 %). Pada program parallel beban prosesnya cukup terbagi rata meskipun terlihat lebih membebankan pada core 2 (30,7%) dan 3 (20,38 %). Penelitian ini dimaksutkan untuk membandingkan gerakan, kecepatan dan akurasi gerak robot lengan dengan menggunakan metode parallel dan sekuensial. Hasil perhitungan akurasi dari kedua metode sama, keduanya sama - sama memiliki nois dan error yang sama yaitu sebesar 0.4 cm dari sumbu x, 1.5 cm dari sumbu y dan 1.06 cm dari sumbu z. Selain itu error juga terjadi pada sudut servo, dimana rata – rata error pada sudut 1 sebesar 1^o, pada sudut 2 sebesar 1.24^o, dan pada sudut 3 sebesar 4.1^o. Bila dibandingkan dari segi kecepatan, metode parallel menjalankan seluruh tugas dengan lebih cepat 20 detik (36,4% dari waktu sequensial) untuk 4 servo dan 10 detik (22,2% dari waktu sekuensial) untuk 3 servo.

Kata kunci: kinematika, robot lengan, *inverse kinematic,* komputasi parallel, *palallel programing*, OpenMP

ABSTRACT

. The method of conventional robotic arm motion is by forward kinematic, which moves by determining the time of deviation of each joint to reach a point. Unlike the forward kinematic, inverse kinematics move by determining the angle required to perform a motion. Forward kinematics is considered as method that utilizes trial and error, this method become less effective because it takes a long time. Meanwhile, inverse kinematic offers more practical solution with more complex computing calculations. With the development of microcontroller complexity on calculation inverse kinematic can be done more quickly by using parallel computing that can take advantage of all the existing cores on the microcontroller. OpenMP is a C ++ program directory that describes the use of application program interfaces to perform parallel computations in multiple memory environments. OpenMP becomes one of the programs that can be utilized to perform parallel computing. The result is, OpenMP able to divide the program so that the Squential program imposes its process on core 1 (30.74%). But in the parallel program the process load fairly even distributed, although it looks more burdensome to core 2 (30,7%) and 3 (20,38%). Accuracy calculation results from both methods are same, both have same nois and error that is equal to 0.4 cm from x axis, 1.5 cm from y axis and 1.06 cm from z axis. In addition, the error also occurs at the servo angle, where the average error in 1st servo is 1^o, at the 2nd servo is 1.24^o, and at the 3rd servo is 4.1^o. When compared to speed, the parallel method performs all tasks faster than squential by 20 seconds (36.4% of the sequential time) for 4 servos and 10 seconds (22.2% of the squared time) for 3 servos.

Keywords: kinematics, robot arm, inverse kinematic, parallel computing, parallel programing, OpenMP

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
KATA PENGANTAR iv
ABSTRAK v
ABSTRACT vi
DAFTAR ISI vii
DAFTAR TABEL x
DAFTAR GAMBAR xi
BAB 1 PENDAHULUAN 1
1.1 Latar belakang 1
1.2 Rumusan masalah 2
1.3 Tujuan 2
1.4 Manfaat 3
1.5 Batasan masalah 3
1.6 Sistematika pembahasan 3
BAB 2 LANDASAN KEPUSTAKAAN 5
2.1 Tinjauan Pustaka 5
2.2 Dasar Teori 6
2.2.1 Robot Manipulator 6
2.2.2 Kinematika Robot 7
2.2.3 *Inverse Kinematic* 8
2.2.4 Komputasi Paralel 8
2.2.5 OpenMP 9
2.2.6 Servo 10
2.2.7 Raspberry Pi 11
BAB 3 METODOLOGI 13
3.1 Studi Literatur 14
3.2 Analisis kebutuhan 14
3.3 Perancangan sistem 14
3.4 Implementasi 15

3.5 Pengujian dan Analisis 15
3.6 Penutup 15
BAB 4 ANALISIS KEBUTUHAN 16
4.1 Gambaran Umum Sistem 16
4.2 Kebutuhan Sistem 16
4.2.1 Kebutuhan Fungsional 16
4.2.2 Kebutuhan Non Fungsional 16
BAB 5 PERANCANGAN DAN IMPLEMENTASI 18
5.1 Perancangan 18
5.1.1 Perancangan Desain Robot 18
5.1.2 Perancangan Rangkaian Elektronika 20
5.1.3 Perancangan Komputasi Paralel 21
5.1.4 Perancangan Inverse Kinematic 23
5.2 Implementasi 23
5.2.1 Implementasi Desain Robot 23
5.2.2 Implementasi rangkaian elektronika 24
5.2.3 Implementasi Komputasi Paralel dan Inverse Kinematic 25
BAB 6 PENGUJIAN DAN ANALISIS 27
6.1 Pengujian CPU Usage 27
6.1.1 Tujuan Pengujian 27
6.1.2 Proses Pengujian 27
6.1.3 Hasil Pengujian 27
6.2 Pengujian waktu 30
6.2.1 Tujuan Pengujian 30
6.2.2 Proses Pengujian 30
6.2.3 Hasil Pengujian 30
6.3 Pengujian akurasi 32
6.3.1 Tujuan Pengujian 32
6.3.2 Proses Pengujian 32
6.3.3 Hasil Pengujian 33
BAB 7 Penutup 36
7.1 Kesimpulan 36

7.2 Saran 37
DAFTAR PUSTAKA..... 38
LAMPIRAN A Source Code Program Paralle..... 39
LAMPIRAN B Source code program sekuensial..... 44

DAFTAR TABEL

Tabel 2.1 spesifikasi servo 11
Tabel 5.1 Tabel pin schematic 20
Tabel 5.2 Potongan Source Code 25
Tabel 6.1 pengujian cpu usage 1 29
Tabel 6.2pengujian cpu usage 2 30
Tabel 6.3 pengujian waktu dengan 4 servo 31
Tabel 6.4 tabel pengujian waktu dengan 3 servo 31
Tabel 6.5 pengujian akurasi titik akhir 33
Tabel 6.6 pengujian akurasi sudut 33
Tabel 6.7 pengujian akurasi 3 34

DAFTAR GAMBAR

Gambar 2.1 Robot biped 5
Gambar 2.2 Robot Lengan 7
Gambar 2.3 posisi *joint* robot 8
Gambar 2.4 Parallelizing OpenMP 9
Gambar 2.5 Servo HS-322HD 10
Gambar 2.6 Raspber*ry Pi 3 tipe B* 12
Gambar 3.1 Diagram Alir Metodologi Penelitian. 13
Gambar 5.1 diagram blok sistem 18
Gambar 5.2 *Joint* yang digunakan 18
Gambar 5.3 Kerangka robot 19
Gambar 5.4 desain robot 19
Gambar 5.5 Schematik rangkaian 20
Gambar 5.6 flow chart sistem 22
Gambar 5.7 Dokumentasi Open MP 23
Gambar 5.8 modifikasi persamaan inverse kinematic 23
Gambar 5.9 hasil potongan bahan 24
Gambar 5.10 rangkaian elektronika 24
Gambar 5.11 robot lengan 25
Gambar 6.1 resource monitor saat program belum berjalan 28
Gambar 6.2 resource monitor program sekuensial 28
Gambar 6.3 resource monitor program dengan parallel 29
Gambar 6.4 perbandingan waktu kerja robot 32

BAB 1 PENDAHULUAN

1.1 Latar belakang

Robot adalah sebuah alat mekanik yang dapat melakukan tugas fisik dengan menggunakan program yang telah dirancang terlebih dahulu (Desiani, 2015). Sedangkan menurut KBBI (Kamus Besar Bahasa Indonesia) robot didefinisikan sebagai alat menyerupai manusia dan sebagainya yang dapat bergerak dan berperilaku seperti manusia yang dikendalikan oleh mesin. Robot merupakan suatu sistem yang dapat melakukan suatu pekerjaan untuk memudahkan pekerjaan manusia. Robot memiliki berbagai macam konstruksi. Diantaranya adalah Mobile Robot yaitu robot dapat bergerak berpindah tempat dan Manipulator Robot berupa robot yang terdiri dari link dan joint. Robot – robot dimanfaatkan diberbagai bidang, seperti kedokteran, industri, pertanian dan militer. Dibidang kedokteran dan industri, Manipulator Robot seringkali dimanfaatkan untuk pencampuran bahan kimia yang dapat membahayakan manusia. Di bidang pertanian beberapa negara maju memanfaatkan mobile robot untuk monitoring dan perawatan tanaman dan lahan pertanian.

Manipulator Robot merupakan robot yang terdiri dari kombinasi link dan joint. Link ialah bagian kaku yang menghubungkan sendi atau poros. Poros yang merupakan komponen penggerak yang menyebabkan gerak relatif dari link (RobotWorx, 2017). Robot manipulator ini sangat bermanfaat dan cukup banyak dipakai dalam bidang industri. Terutama dalam bagian packing dalam suatu pabrik. Lengan robot banyak digunakan pada industri, khususnya industri yang memerlukan ketepatan gerak dan bekerja secara berulang – ulang. Ketepatan gerak merupakan fungsi utama suatu robot manipulator, barang yang akan dipindahkan dari satu titik ke titik lain harus terjangkau ruang gerak robot. Namun gerak robot ke posisi yang tepat membutuhkan komputasi yang kompleks.

Pada robot manipulator, bagian poros atau penggerak menggunakan suatu aktuator. Pada penelitian ini, peneliti menggunakan aktuator berupa motor servo. Untuk menggerakkan servo, sebuat mikrokontroler mengirimkan sinyal PWM (Pulse-Width Modulation). Rangkaian elektronik didalam servo kemudian menterjemahkan sinyal tersebut menjadi sudut putar servo. Ketika servo diperintahkan untuk memutar maka motor akan aktif dan bergerak sampai potensiometer mencapai posisi yang diperintahkan. Servo merupakan bagian yang menentukan pergerakan(motion) pada robot manipulator.

Metode konvensional untuk pergerakan lengan robot adalah dengan menentukan simpangan dan pewaktuan masing – masing sendi untuk membuat robot bergerak. Untuk membuat suatu gerakan dilakukan trial and error untuk menentukan masing – masing PWM dan waktu gerak antar servo. Metode ini tidak memerlukan komputasi rumit, tetapi tidak fleksibel untuk membuat gerakan yang baru atau beragam serta lama prosesnya. Untuk menggerakkan lengan robot dengan lebih fleksibel diperlukan beberapa algoritma untuk mengendalikan servo sebagai penggerak sendi. Salah satu algoritma yang digunakan adalah kinematika

balik (inverse kinematic). Metode *inverse kinematics* merupakan metode pergerakan robot lengan dimana variabel yang diketahui adalah titik koordinat tujuan, sehingga dibutuhkan perubahan besar sudut pada masing-masing *joint* robot lengan agar robot lengan mampu mencapai titik tersebut (Setiawan, 2014). Dengan metode ini mikrokontroler akan melakukan perhitungan matematis yang kompleks untuk menentukan gerakan robot.

Dengan semakin berkembangnya teknologi mikrokontroler, maka memungkinkan digunakannya komputasi paralel untuk meringankan kerja komputasi suatu sistem yang sering kali hanya berpusat pada satu kontroler saja. Bila suatu sistem yang kompleks dikerjakan oleh satu kontroler saja, konsekuensi yang harus diterima adalah waktu komputasi yang lama. Komputasi paralel adalah bentuk komputasi di mana banyak instruksi yang dijalankan secara simultan (diwaktu yang bersamaan) (Dingju Zhu, 2008). .

Berdasarkan latar belakang yang telah dibuat, sistem gerak robot dengan menggunakan inverse kinematic dapat dipercepat waktu kerjanya dengan menggunakan metode perhitungan parallel karena variabel variabel sudutnya tidak saling bergantung atau biasa disebut dengan variabel bebas. Dengan diterapkanya kedua metode ini diharapkan robot dapat bergerak dengan akurasi yang tinggi dan waktu komputasi yang cepat.

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, maka dapat dirumuskan permasalahan berikut.

1. Bagaimana cara merekayasa dan mengimplementasikan pola bergerak lengan Robot dengan menggunakan metode *Inverse Kinematic* ?
2. Bagaimana membagi tugas komputasi sudut hasil perhitungan *Inverse Kinematic* dengan menggunakan komputasi parallel pada tiap-tiap *core* ?
3. Bagaimana mengimplementasikan komputasi paralel pada *Inverse Kinematic* dengan menggunakan Raspberry Pi ?

1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Dapat mengimplementasikan metode *Inverse Kinematic* dalam bidang Robotika khususnya pada lengan robot manipulator dengan halus, presisi, dan cepat.
2. Mampu membagi tugas komputasi sudut hasil perhitungan *Inverse Kinematic* dengan menggunakan komputasi parallel pada *core*.
3. Mampu mengimplementasikan komputasi paralel pada *Inverse Kinematic* dengan menggunakan Raspberry Pi.

1.4 Manfaat

1. Bagi Peneliti

   Penelitian ini diharapkan mampu meningkatkan kualitas pengetahuan peneliti dalam bidang keilmuan yang kedepannya dapat digunakan untuk terjun didunia otomasi robotika.

2. Bagi Universitas Brawijaya

   Diharapkan penelitian ini bermanfaat untuk peneltian selanjutnya dalam segmen dan topik yang berbeda.

1.5 Batasan masalah

Agar permasalahan yang telah dirumuskan tidak melebar dari topik, maka penelitian ini dibatasi dalam hal :

1. Penelitian difokuskan pada lengan robot manipulator tanpa badan dengan jumlah total 4 servo.
2. Penelitian difokuskan pada perhitungan gerak robot dari 1 titik ke titik lain.
3. Maksimal putaran servo hanya sebesar 200 derajat

1.6 Sistematika pembahasan

Laporan penelitian ini dibagi atas beberapa bagian.

BAB I PENDAHULUAN.

Pada bagian ini diuraikan latar belakang penelitian, permasalahan-permasalahan yang tercakup pada penelitian, tujuan penelitian, manfaat yang dapat diambil dari penelitian, serta batasan dan ruang lingkup dari penelitian. Dan dibagian akhir diuraikan sistematikan penyajian laporan penelitian.

BAB II LANDASAN KEPUSTAKAAN.

Pada bagian ini dipaparkan teori-teori serta pustaka yang dipakai pada saat melakukan penelitian. Teori-teori ini diambil dari buku literatur dan dari internet. Teori yang dibahas meliputi teori tentang penelitian tentang robotika dan komputasi paralel.

BAB III METODOLOGI PENELITIAN

Bagian ketiga memaparkan langkah-langkah yang digunakan untuk membahas permasalahan dalam penelitian. Pada bagian ini dijelaskan alat dan metode yang digunakan untuk melakukan perencanaan dan mendapatkan spesifikasi kebutuhan robot. Selain itu dipaparkan juga metode yang digunakan untuk merancang dan menganalisa sistem.

BAB IV ANALISIS KEBUTUHAN

Pada bab ini membahas kebutuhan – kebutuhan yang diperlukan agar sistem dapat berjalan. Kebutuhan tersebut melingkupi kebutuhan fungsional dan nonfungsional.

BAB V PERANCANGAN DAN IMPLEMENTASI

Pada bab ini membahas perancangan desain robot lengan, perancangan elektronika yang digunakan, serta perancangan algoritma *worksharing*. Pada bagian akhir dari perancangan ini akan mendapat hasil rancangan yang kemudian diimplementasikan.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini membahas proses dan hasil pengujian terhadap sistem yang telah direalisasikan dan kesesuaian dengan yang diharapkan.

BAB VII KESIMPULAN DAN SARAN

Pada bab ini membahas tentang kesimpulan dan saran sesuai hasil penelitian berdasarkan uraian-uraian dari bab-bab sebelumnya untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Pada dasarnya telah terdapat beberapa penelitian yang memiliki keterkaitan dengan penelitian ini, keterkaitan yang ada berupa kesamaan teknologi yang digunakan, metode dan environment. Berikut beberapa penelitian yang memiliki keterkaitan dengan penelitian ini.

Penelitian berjudul “Aplikasi Sensor Proximity Pada Lengan Robot Sebagai Penyortir Kotak Berdasarkan Ukuran Berbasis Arduino Uno“ dilakukan oleh Desiani pada tahun 2015. Tujuan penelitian tersebut adalah untuk mempelajari cara kerja Sensor Proximity dalam mendeteksi 3 kotak yang berbeda ukuran yakni kotak kecil, kotak sedang dan kotak besar. Lengan robot memindahkan kotak yang telah dideteksi ke suatu tempat sesuai ukuranya. Robot lengan yang digunakan menggunakan robot lengan 3 DOF. Kelebihan robot ini, karena posisi benda statis robot tidak memerlukan komputasi yang kompleks. Dengan metode trial dan error mikrokontroler tidak perlu melakukan komputasi algoritma sehingga tidak membebani mikrokontroler. Namun bila posisi benda yang akan di pilah tidak statis maka robot tersebut perlu diberikan algoritma tambahan yang akan membebani mikrokontroler

Penelitian berjudul “Penerapan Invers Kinematika Untuk Pergerakan Kaki Robot Biped” dilakukan oleh Surya Setiawan, Budi Rahmadya, Derisma dan Firdaus. Tujuan penelitian tersebut adalah untuk menerapkan metode invers kinematika terhadap gerakan kaki robot biped agar dapat berjalan. Robot biped ini menggunakan 6 buah motor servo pada kedua kakinya (3 kanan dan 3 kiri) seperti pada Gambar 2.1.

Gambar 2.1 Robot biped

Sumber : (Setiawan, et al., 2015)

Penelitian tersebut menghasilkan rumus perhitungan inverse kinematic untuk posisi kaki ke depan, tegak dan ke belakang untuk masing – masing kaki. Keunggulan inverse kinematic berdasarkan penelitian tersebut menunjukan bahwa metode ini sangat fleksibel karena dengan menentukan suatu kordinat yang dituju oleh End Effector (ujung) dalam kasus ini telapak kaki. Peneliti dapat memperhitungkan besar sudut putar masing – masing servo. Kelemahan dari penelitian tersebut, berdasarkan percobaan yang dilakukan, terdapat 2 – 5% error dari masing-masing sudut pada kaki robot biped.

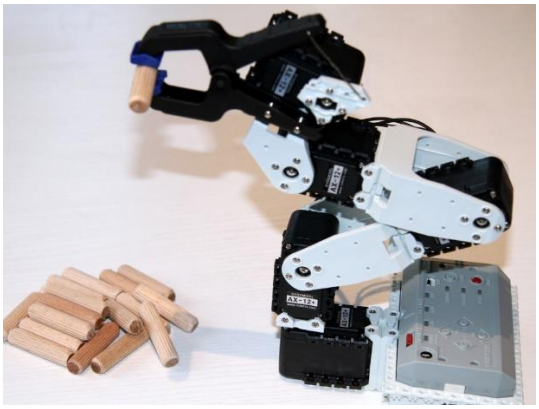
Penelitian berjudul "Application of Parallel Computing in Digital City" menjelaskan tentang implementasi komputasi parallel dalam bidang picture recognition. Pada suatu kota objek objek didalamnya mengalami event event tertentu yang terkadang terjadi secara bersamaan. Misal, saat terjadi penebangan pohon di suatu wilayah, terjadi kebakaran diwaktu yang sama. Oleh karena itu Digital City ini memerlukan perhitungan pada objek objeknya (Building, pohon, dan rumput) secara simultan. Dari penelitian tersebut dijelaskan bahwa hasil perhitungan dengan komputasi parallel sangat efektif untuk perhitungan dengan skala besar dan real time.

Berdasarkan penelitian pertama masalah yang dihadapi adalah fleksibilitas gerak dan kompleksitas komputasi. Olehkarena itu peneliti mencoba menawarkan solusi berupa penambahan metode perhitungan Inverse Kinematic dengan memanfaatkan komputasi paralel untuk perhitunganya. Inverse kinematic seperti yang di jelaskan pada penelitian kedua, berfungsi untuk memberi gerak yang lebih fleksibel pada robot. Pada penelitian ketiga membahas tentang komputasi parallel yang diimplementasikan pada Digital City, sedangkan peneliti ingin mengimplementasikan komputasi parallel ini pada perhitungan Inverse Kinematic lengan robot manipulator. Diharapkan dengan menerapkan metode komputasi parallel dapat mempercepat kerja alat secara efektif.

2.2 Dasar Teori

2.2.1 Robot Manipulator

Manipulator Robot merupakan robot yang terdiri dari kombinasi link dan joint. Link adalah bagian kaku yang menghubungkan sendi atau poros. Poros adalah komponen penggerak yang menyebabkan gerak relatif dari link (RobotWorx, 2017). Robot lengan adalah salah satu jenis robot manipulator. Robot lengan dapat dilihat pada Gambar 2.2.



Gambar 2.2 Robot Lengan

Sumber : (RobotFreak, 2017)

Robot lengan memiliki beberapa bagian yaitu:

1. *Link*: bagian robot yang bentuknya tetap dan dapat bergerak. Link biasanya dihubungkan dengan joint.
2. *Joint*: penghubung link dengan link atau base yang dapat bergerak aktif (biasanya terdapat aktuator). Pada robot lengan, joint ini adalah motor servonya.
3. *End Effector* (ujung): titik akhir yang menghubungkan robot manipulator dengan objek. *End effector* pada robot lengan terletak dibagian ujung dari lengan robot yang menyentuh langsung dengan objek kerja lengan robot.
4. DoF *(Degree of Freedom*): jumlah gerakan independen yang dapat dilakukan oleh suatu robot.

2.2.2 Kinematika Robot

Kinematika dalam robot adalah suatu bentuk pernyataan yang berisi tentang deskripsi matematik geometri dari suatu struktur robot. Dari persamaan kinematik dapat diperoleh hubungan antara konsep geometri ruang sendi pada robot dengan konsep koordinat yang bisa dipakai untuk menentukan kedudukan dari suatu obyek. (Maulana, 2015). Pergerakan sebuah robot selalu berkaitan dengan workspace atau ruang kerjanya, ruang kerja robot merupakan seluruh lokasi yang mampu dijangkau oleh ujung dari robot manipulator. Kinematika robot dapat didefinisikan menjadi dua yaitu inverse kinematics dan forward kinematics. Forward Kinematics atau kinematika maju merupakan perhitungan dalam menentukan posisi ujung robot manipulator terhadap koordinat berdasarkan perubahan gerak pada masing – masing joint yang menyusunnya (Setiawan, 2014). Sedangkan *inverse kinematics* adalah proses penentuan perubahan gerak yang harus dilakukan oleh masing - masing joint pada lengan robot untuk dapat mencapai suatu titik koordinat tertentu. Penelitian ini difokuskan pada pergerakan robot lengan menggunakan metode inverse kinematics.

2.2.3 *Inverse Kinematic*

Inverse kinematic merupakan suatu metode analisa untuk melakukan transformasi dari ruang *Cartesian* ke ruang sendi. Metode ini menentukan nilai sudut berdasarkan titik *end effector*. Dalam menentukan koordinat *end effector* harus disesuaikan dengan batas area kerja dari jangkauan robot. Secara matematis *inverse kinematic* dapat ditentukan dengan persamaan 2.1 – 2.5 :

$$\boldsymbol{\theta_1 \leftarrow atan2(y, x)} \quad \textbf{2.1}$$

$$\boldsymbol{c_3 \leftarrow \frac{x^2+y^2+(z-l_1)^2-l_2^2-l_3^2}{2l_2l_3}} \quad \textbf{(2.2)}$$

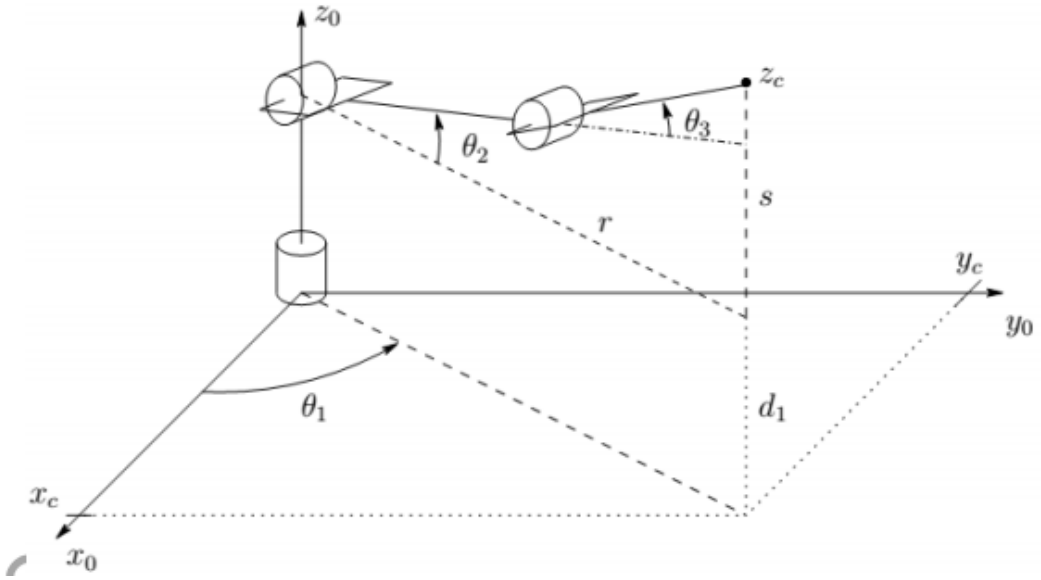
$$\boldsymbol{s_3 \leftarrow +\sqrt{1-c_3^2}} \quad \textbf{(2.3)}$$

$$\boldsymbol{\theta_3 \leftarrow atan2(s_3, c_3)} \quad \textbf{(2.4)}$$

$$\boldsymbol{\theta_2 \leftarrow atan2\left(z-l_1, \sqrt{x^2+y^2}\right) - atan2(l_3s_3, l_2+l_3c_3)} \quad \textbf{(2.5)}$$

Sumber : Lecture Notes Prepared by Daniela Rus EECS/MIT Spring (Rus, 2011)

Dengan syarat model robot menggunakan 3 dof *cylindrical joint* dengan peletakan seperti pada Gambar 2.3.



Gambar 2.3 posisi *joint* robot

Sumber : Robot Modeling and Control First Edition (Mark W. Spong, 2005)

Sedangkan untuk lengan robot dengan 4 dof. Terdapat perubahan persamaan berdasarkan lokasi dof ke 4. Perhitungan dan desain 4 dof akan dibahas pada bab perancangan.

2.2.4 Komputasi Paralel

Komputasi paralel adalah suatu teknik untuk mengerjakan suatu perhitungan yang besar dan memerlukan waktu yang lama dengan memecah

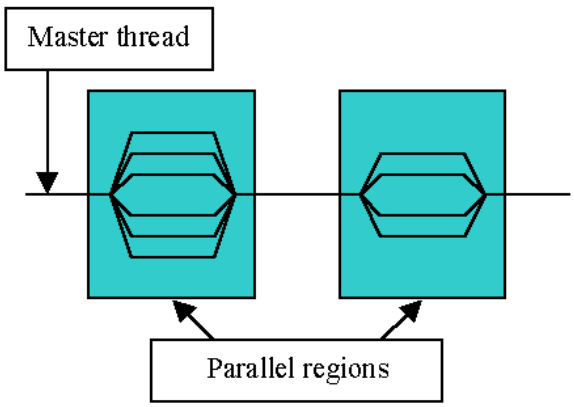
menjadi beberapa bagian dan mengerjakannya secara independen di prosesor yang berbeda (Zarkasih, et al., 2013). Salah satu teknologi yang telah dikembangkan adalah teknologi antar muka yang disebut OpenMP. OpenMP adalah direktori program C++ yang menggambarkan penggunaan antarmuka program aplikasi untuk melakukan perhitungan paralel di lingkungan memori secara bersamaan (Peter Arbenz, 2011). OpenMP memungkinkan pengguna menandai area kode, seperti, *while* atau *loop*, yang sesuai untuk pemrosesan paralel. Pengguna dapat memberitahu *compiler* untuk memperhatikan program akan dibuat yang berjalan secara paralel dan serial (*squential*).

OpenMP cocok untuk sistem paralel *shared memory*, yaitu situasi di mana ada satu ruang memori tunggal, dan beberapa prosesor. Jika memori dibagi, maka biasanya jumlah prosesornya akan kecil dan program akan berjalan pada *physical memory* yang sama.

2.2.5 OpenMP

OpenMP (*Open Multi-Processing*) adalah sebuah API (*application programming interface*) yang mendukung multi processing shared memory. OpenMP dikelola oleh konsorsium teknologi nonprofit atau *opensource* yang bernama *OpenMP Architecture Review Board* (OpenMP ARB), yang didefinisikan bersama oleh sekelompok vendor perangkat keras dan perangkat lunak komputer utama, termasuk AMD, IBM, Intel, Cray, HP, Fujitsu, Nvidia, NEC, Red Hat , Texas Instruments, Oracle Corporation, dan banyak lagi. OpenMP Architecture Review Board (ARB) menerbitkan spesifikasi API pertama, OpenMP untuk Fortran 1.0, pada bulan Oktober 1997. Pada bulan Oktober tahun berikutnya mereka merilis untuk C / C ++. Hingga saat penelitian ini ditulis OpenMP sudah merilis hingga versi 4.5.

OpenMP menggunakan model portabel yang memberikan *developer* sebuah antarmuka yang sederhana dan fleksibel untuk mengembangkan aplikasi paralel pada komputer desktop standar hingga superkomputer.



Gambar 2.4 Parallelizing OpenMP

Sumber :
https://www.dartmouth.edu/~rc/classes/intro_openmp/print_pages.shtml

Berdasarkan Gambar 2.4, OpenMP menggunakan metode *parallelizing* dimana thread master (serangkaian instruksi yang dijalankan secara berurutan) menghasilkan sejumlah *slave threads* dan sistem membagi tugas di antara mereka. Thread kemudian berjalan bersamaan dengan mengalokasikan masing masing thread ke prosesor yang berbeda. Thread inilah yang dideklarasikan sebagai Section pada code program yang akan dibuat.

2.2.6 Servo

Aktuator merupakan penggerak robot, yang dimaksud dalam lengan robot dalam penelitian ini adalah seluruh unit yang mengendalikan system rangka pasif layaknya otot sendi manusia. Karena rangka bersifat pasif atau tidak dapat bergerak maka harus ada penggeraknya yang dapat mengontrol rangka untuk dapat bergerak dan berpindah ke suatu titik. Aktuator yang digunakan adalah servo dengan tipe HS-322HD buatan Hi-Tech. Motor servo merupakan sebuah perangkat yang dapat berputar (motor) yang dirancang dengan sistem kontrol umpan balik loop tertutup (servo), sehingga dapat di set-up atau di atur untuk menentukan dan memastikan posisi sudut dari poros output motor. Motor servo merupakan perangkat yang terdiri dari motor DC, serangkaian gear, rangkaian kontrol dan potensiometer. Serangkaian gear yang melekat pada poros motor DC akan memperlambat putaran poros dan meningkatkan torsi motor servo, sedangkan potensiometer dengan perubahan resistansinya saat motor berputar berfungsi sebagai penentu batas posisi putaran poros motor servo.

Berikut gambar servo tipe HS-322HD yang dipakai :

Gambar 2.5 Servo HS-322HD

Sumber : http://www.jameco.com/Jameco/Products

Tabel 2.1 spesifikasi servo

Dimensions	1.57" x 0.78" x 1.43" (39.88 x 19.81 x 36.32mm)
Product Weight	1.52oz (43g)
Voltage Range	4.8V - 6.0V
No-Load Speed (4.8V)	0.19sec/60°
No-Load Speed (6.0V)	0.15sec/60°
Stall Torque (4.8V)	42 oz/in (3.0 kg/cm)
Stall Torque (6.0V)	51 oz/in (3.7 kg/cm)
Max PWM Signal Range (Standard)	553-2450µsec
Max Travel (out of box)	201°
Pulse Amplitude	3-5V
Operating Temperature	-20°C to +60°C
Current Drain - no-load (4.8V)	160mA
Current Drain - no-load (6V)	180mA

Berdasarkan tabel Tabel 2.1, servo Hi-Tech HS-322HD bekerja pada tegangan 4,8 hingga 6 Volt dengan torsi sebesar 3,0 hingga 3,7 kg. Maksimal sudut yang dapat dikerjakan servo adalah 201°.

2.2.7 Raspberry Pi

Raspberry Pi adalah modul mikrokontroler yang juga mempunyai *input output digital port* seperti pada *board microcontroller*. Diantara kelebihan Rasberry Pi dibanding board microcontroller yg lain mempunyai Port/koneksi untuk display berupa TV atau Monitor PC serta koneksi *USB* untuk Keyboard serta Mouse. *Operating system* yang banyak dipakai antara lain Linux Raspbian. OS disimpan di *SD card* dan saat proses *boot* OS hanya bisa dari *SD card* tidak dari lokasi lain.

Gambar 2.6 Raspber*ry Pi 3 tipe B*

Sumber : https://cdn.shopify.com/s/files/

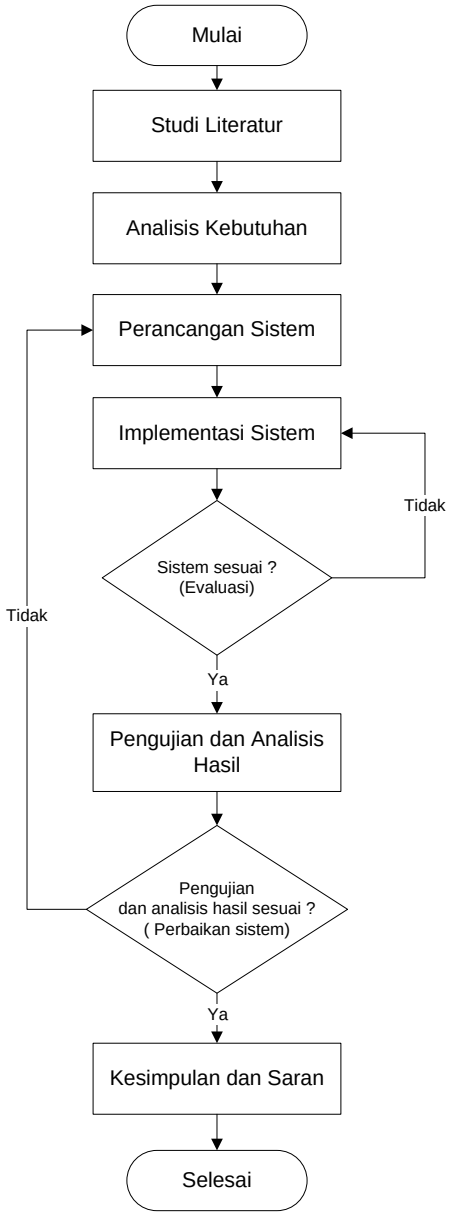
Adapun spesifikasi dari raspberry pi 3 tipe ada adalah sebagai beriku :

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

Raspberry Pi 3 adalah generasi ketiga Raspberry Pi. Ini menggantikan Raspberry Pi 2 Model B pada bulan Februari 2016. Pada dasarnya raspberry pi tipe 2 dan 3 tidak memiliki perbedaan yang besar, raspberry pi 3 sudah memiliki modul wifi *onboard* sedangkan raspberry pi 2 tidak. Dalam penelitian ini raspberry pi dipilih sebagai kontroler karena raspberry pi memiliki prosesor dengan 4 buah core. Raspberry yang digunakan tidak spesifik harus menggunakan raspberry pi 3, semua jenis raspberry dapat digunakan asalkan sudah berspesifikasi *Multi Core*.

BAB 3 METODOLOGI

Dari latar belakang yang telah dibuat, penelitian ini difungsikan untuk memberikan gambaran dalam merekayasa lengan robot manipulator dengan metode gerak *inverse kinematic.* Selain itu ditampilkan juga perbandingan penggunaan metode komputasi secara parallel dengan sekuensial. Gambaran umum tahapan metodolgi penelitian disajikan dalam diagram alir pada Gambar 3.1



Gambar 3.1 Diagram Alir Metodologi Penelitian.

3.1 Studi Literatur

Studi Literatur berisi kajian - kajian yang sebagai dasar penelitian melingkupi penelitian – penelitian dan teori yang telah berkembang sebelumnya, sesuai kebutuhan penelitian penulis. Adapun teori-teori yang dikaji adalah sebagai berikut:

1. Robot Lengan 4 DOF (Degree of freedom)
2. Inverse Kinematics
3. Komputasi Parallel
4. Raspberry Pi
5. Motor Servo Hi-Tech

3.2 Analisis kebutuhan

Didalam proses analisis kebutuhan sistem akan dijelaskan kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional menggambarkan kebutuhan utama sistem yang menjadi kriteria utama penelitian. Sedangkan kebutuhan non fungsional merupakan kebutuhan batasan sistem agar sistem dapat bekerja dengan baik dan tidak mengganggu kebutuhan fungsional.

Kebuthan fungsional sistem terdiri dari :

1. Robot dapat bergerak ke titik yang dituju dengan akurat
2. Mikrokomputer dapat mengirimkan perintah gerakan ke masing-masing servo
3. Minikomputer dapat menjalankan program secara parallel dan sekuensial

Kebutuhan non-fungsional sistem terdiri dari :

1. Tegangan atau power yang diberikan harus sesuai dengan yang dibutuhkan
2. Penelitian difokuskan pada lengan robot manipulator tanpa badan dengan jumlah total 4 servo.
3. Penelitian difokuskan pada perhitungan gerak robot dari 1 titik ke titik lain.
4. Maksimal putaran servo hanya sebesar 200 derajat.

3.3 Perancangan sistem

Perancangan sistem dilakukan sebelum dilakukan kegiatan implementasi agar penelitian dapat dilakukan dengan efektif dan terfokus. Persiapan pertama yaitu membuat diagram blok sistem yang nantinya akan digunakan untuk perancangan sistem sebagai alat baca rancangan yang akan dibuat. Kedua, dilakukan perancangan elektronika sistem. Dan yang ketiga yaitu adanya alur program yang terdiri dari perintah untuk menjalankan suatu sistem sebagai gambarannya.

3.4 Implementasi

Implementasi meliputi proses perancangan sistem penelitian sampai dengan pada hasil akhir penelitian. Adapun tahapan dari implementasi sistem pada penelitian ini adalah implementasi desain robot, rangkaian elektronika, serta algoritma *worksharing.*

3.5 Pengujian dan Analisis

Pengujian dan analisis hasil dilakukan untuk mengetahui apakah sistem berjalan dengan baik dan sesuai dengan spesifikasi kebutuhan yang telah ditetapkan. Pengujian dan analisis dilakukan melalui 3 cara. Membandingkan waktu proses komputasi pada lengan robot saat menggunakan metode komputasi paralel dengan saat tidak menggunakan metode komputasi paralel. Membandingkan besarnya CPU *usage* saat program komputasi pada lengan robot dijalankan menggunakan metode parallel dengan saat program komputasi lengan robot dijalankan dengan metode sekuensial. Menghitung akurasi pergerakan servo dengan melihat seberapa besar error yang terjadi.

3.6 Penutup

Dari hasil pengujian dan analisis kemudian diambil kesimpulan untuk menentukan hasil penelitian. Dimana kesimpulan merupakan jawaban dari rumusan masalah yang telah dibuat berdasarkan hasil analisis dari pengujian yang telah dibuat. Pada bab ini disajikan pula saran dari peneliti yang berisi kesulitan dan pengembangan yang dapat dilakukan dari penelitian ini.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Komputasi parallel merupakan teknologi yang berguna untuk mempercepat suatu perhitungan yang kompleks. Komputasi paralel membagi perhitungan dalam suatu sistem menjadi beberapa bagian untuk dihitung secara bersamaan. Salahsatu perhitungan yang dapat diolah menggunakan komputasi parallel adalah inverse kinematic. Inverse kinematik melakukan perhitungan untuk tiap tiap sendi yang ada pada robot manipulator. Sehingga untuk masing masing sendi robot dapat dihitung secara terpisah.

4.2 Kebutuhan Sistem

Pada bagian ini berisi penjelasan kebutuhan fungsional, kebutuhan non fungsional, kebutuhan perangkat keras dan kebutuhan perangkat lunak yang dianalisis sesuai dengan apa yang sistem butuhkan sehingga diharapkan dapat mempermudah dalam melakukan desain sistem dan implementasi sistem.

4.2.1 Kebutuhan Fungsional

Adapun kebutuhan fungsional yang harus dipenuhi dalam penelitian ini antara lain :

1. **Robot dapat bergerak ke titik yang dituju dengan akurat**

   End efector robot dapat bergerak ke titik yang dituju dengan lebih cepat dengan akurasi yang tinggi

2. **Mikrokomputer dapat mengirimkan perintah gerakan ke masing-masing servo**

   Mikrokomputer mengirimkan perintah ke masing masing servo secara parallel dan sekuensial. Dari 4 servo yang ada, secara sekuensial servo bergerak bergantian. Secara parallel servo akan bergerak segera setelah nilai sudut ditemukan tanpa menunggu servo sebelumnya selesai bergerak.

3. **Minikomputer dapat menjalankan program secara parallel dan sekuensial**

   Raspberry pi dapat mengimplementasikan komputasi paralel dan sekuensial. Ditunjukkan dengan menampilkan perbedaan resorce monitor pada Raspberry pi.

4.2.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional menjelaskan mengenai apa saja yang menjadi batasan terhadap kebutuhan perancangan sistem. Adapun kebutuhan non fungsional yang harus dipenuhi dalam penelitian ini adalah :

1. **Tegangan atau power yang diberikan harus sesuai dengan yang dibutuhkan**

Tegangan yang dibutuhkan Raspberry pi 3 sebesar 5 volt. Servo harus mendapatkan tegangan dari power source tersendiri, bukan dari VCC raspberry pi agar tidak mempengaruhi tegangan pada servo.

2. **Penelitian difokuskan pada lengan robot manipulator tanpa badan dengan jumlah total 4 servo.**

Penelitian ini difokuskan untuk melakukan perhitungan *inverse kinematic* pada robot lengan 4 dof. Servo yang digunakan berjumlah 4 buah.

3. **Penelitian difokuskan pada perhitungan gerak robot dari 1 titik ke titik lain.**

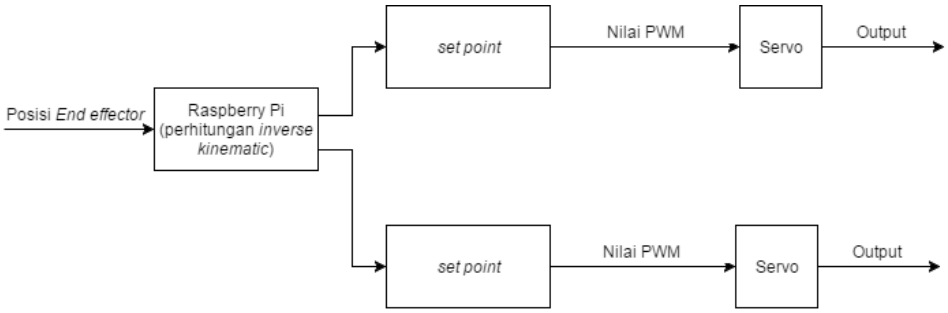
Penelitian ini hanya menggunakan 2 buah titik sebagai acuan perhitungan *inverse kinematic.* Gerakan robot untuk kedua titik tersebut dilakukan secara sekuensial dan secara parallel untuk kemudian dibandingkan akurasi dan kecepatan penyelesaian tugasnya.

4. **Maksimal putaran servo hanya sebesar 200 derajat.**

Servo yang digunakan adalah servo berjenis Hi-Tech dengan tipe HS-322HD dan HS-645MG. Servo dengan jenis ini hanya dapat berputar sebesar 200 derajat.

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini membahas perancangan dan implementasi diantaranya adalah perancangan dan implementasi desain robot,rangkaian elektronika, serta algoritma worksharing. Akhir dari bab ini, penelitian dapat menghasilkan suatu sistem sesuai dengan harapan dan dapat diuji berdasarkan bidang keilmuannya.

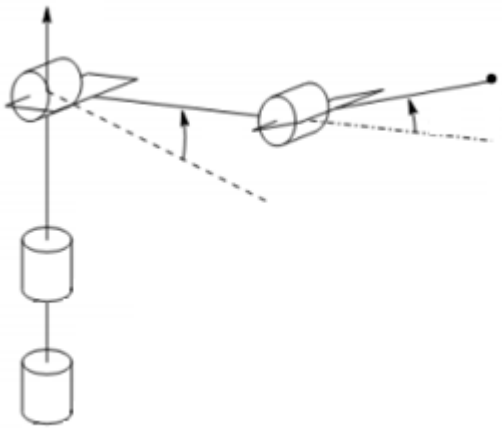


Gambar 5.1 diagram blok sistem

Gambar 5.1 menunjukan bahwa tugas perhitungan inverse kinamtic dijalankan oleh raspberry pi. *Set point* merupakan nilai PWM yang didapat setelah inverse kinematic dihitung. Dalam hal ini *set point* adalah nilai derajat putaran yang harus dicapai servo.

5.1 Perancangan

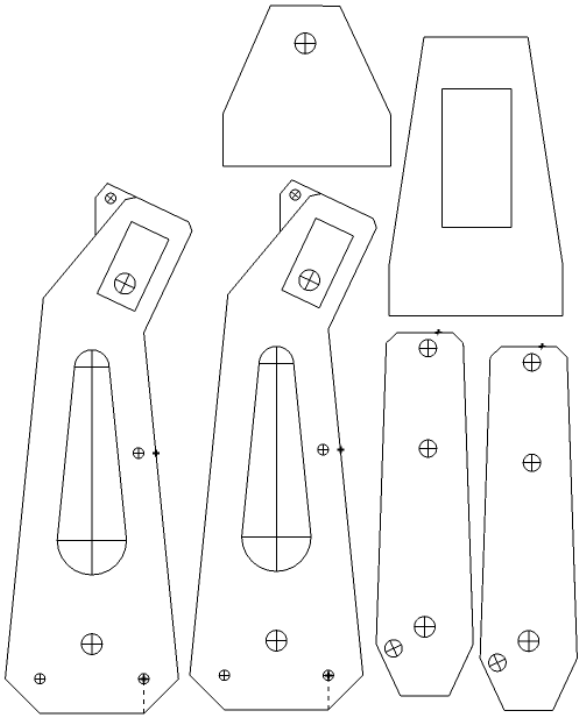
5.1.1 Perancangan Desain Robot



Gambar 5.2 *Joint* yang digunakan

Pada penelitian ini, robot menggunakan 4 buah *revolute joint* dengan posisi seperti tampak pada Gambar 5.2 . *Revolute Joint* memungkinkan komponen untuk berotasi mengelilingi suatu joint yang telah ditetapkan. *Revolute joint* hanya

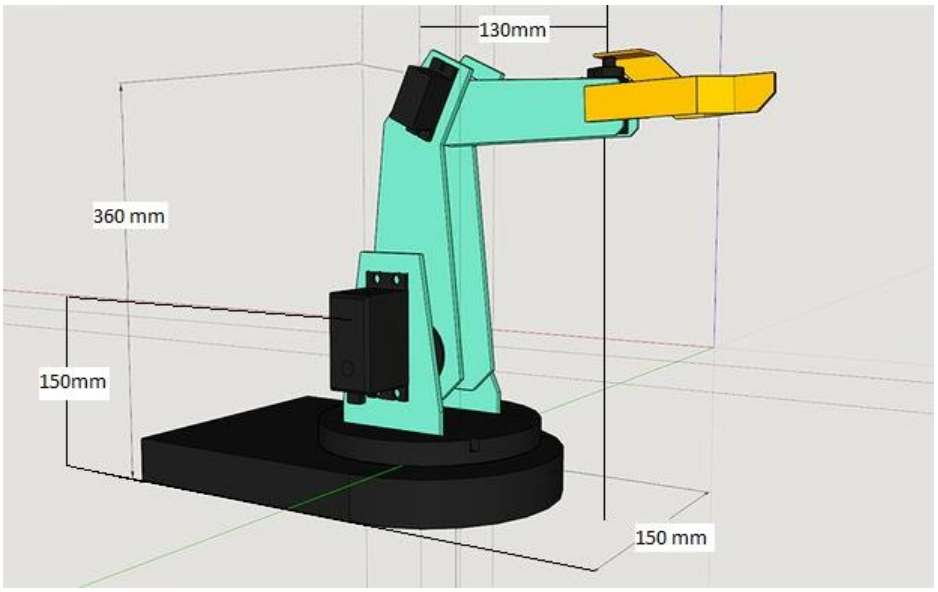
mendapatkan satu gerakan rotasi. Dengan *joint* tidak ada gerak tranlasi tambahan yang diperbolehkan.



Gambar 5.3 Kerangka robot

Desain robot lengan terdiri dari 4 buah servo cylindrical. Kerangka robot terbuat dari bahan Acrylic. Dengan bentuk kurang lebih sesuai dengan

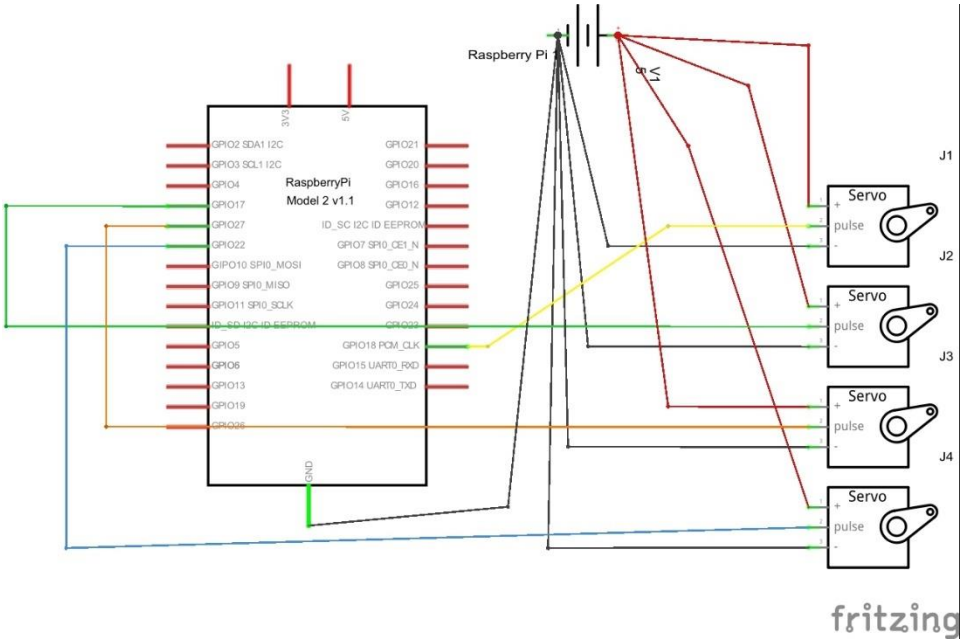
Gambar **5.3**. Bagian bagian tersebut kemudian dirangkai sehingga membentuk lengan robot seperti pada Gambar 5.4.



Gambar 5.4 desain robot

Bagian yang berwarna biru adalah bagian lengan sedangan bagian berwarna kuning meruakan bagian tangan. Bagian yang berbentuk kotak berwarna hitam adalah bagian diletakkanya servo.

5.1.2 Perancangan Rangkaian Elektronika



Gambar 5.5 Schematik rangkaian

BerdasarkanGambar 5.5 kemudian dibuatlah tabel pin agar memudahkan dalam pembacaan rangkaian shcematic.

Tabel 5.1 Tabel pin schematic

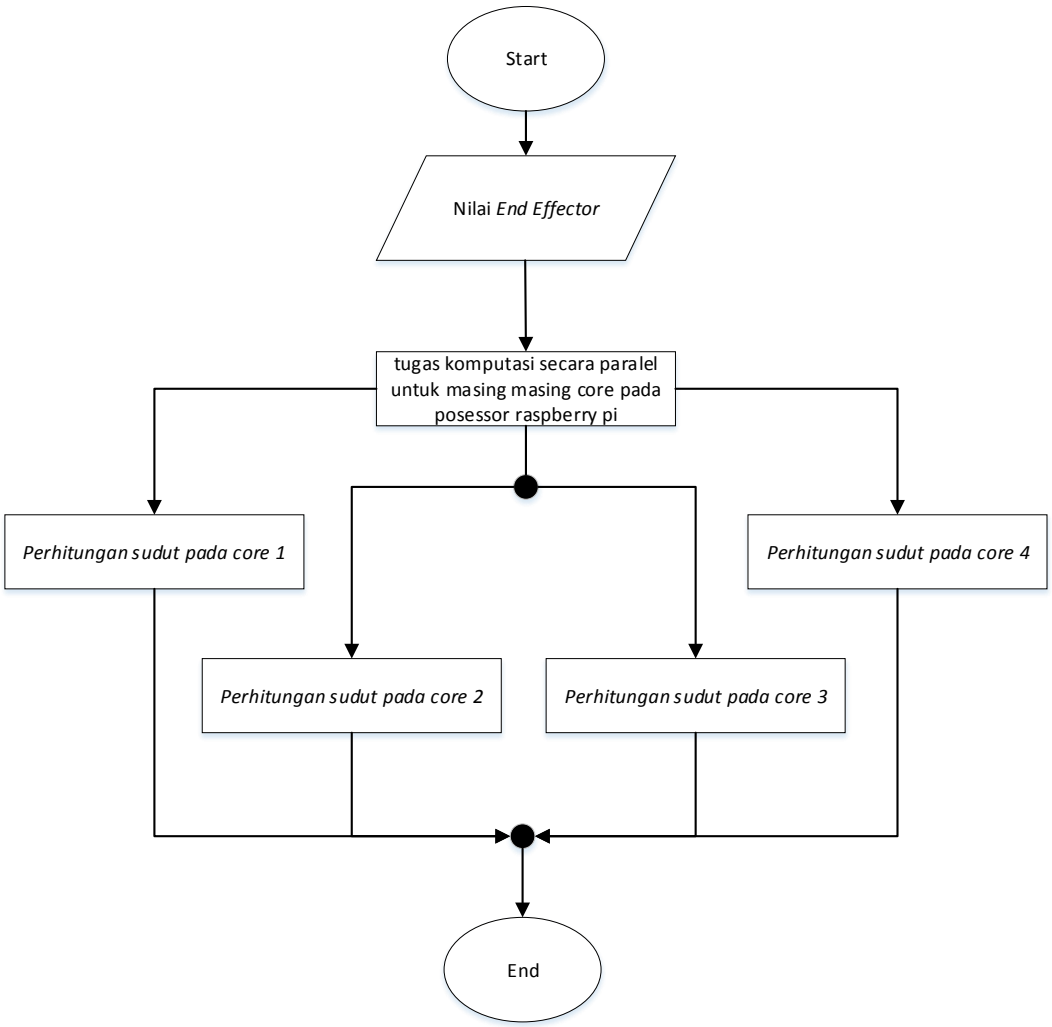
Servo 1	Servo 2	Servo 3	Servo 4	Pwr DC 5V	Raspberry
VCC	VCC	VCC	VCC	5V	-
GND	GND	GND	GND	GND	GND
Pulse	-	-	-	-	GPIO 17
-	Pulse	-	-	-	GPIO 18
-	-	Pulse	-	-	GPIO 27
-	-	-	Pulse	-	GPIO 22

Tabel 5.1 menunjukan tabel pin berdasarkan rancangan schematic yang telah dibuat. Seluruh VCC dari servo memperoleh tegangan dari power source 5v, hal ini dikarenakan tegangan servo sebesar 4,8v – 6v. Bila VCC servo dihubungkan dengan VCC raspberry maka saat servo bergerak tegangan raspberry akan turun dan menyebabkan raspberry mengalami *reboot.* Pin *pulse* dari servo di hubungkan dengan pin GPIO sesuai tabel, GPIO 17 sama dengan pin fisik 11, GPIO 18 sama

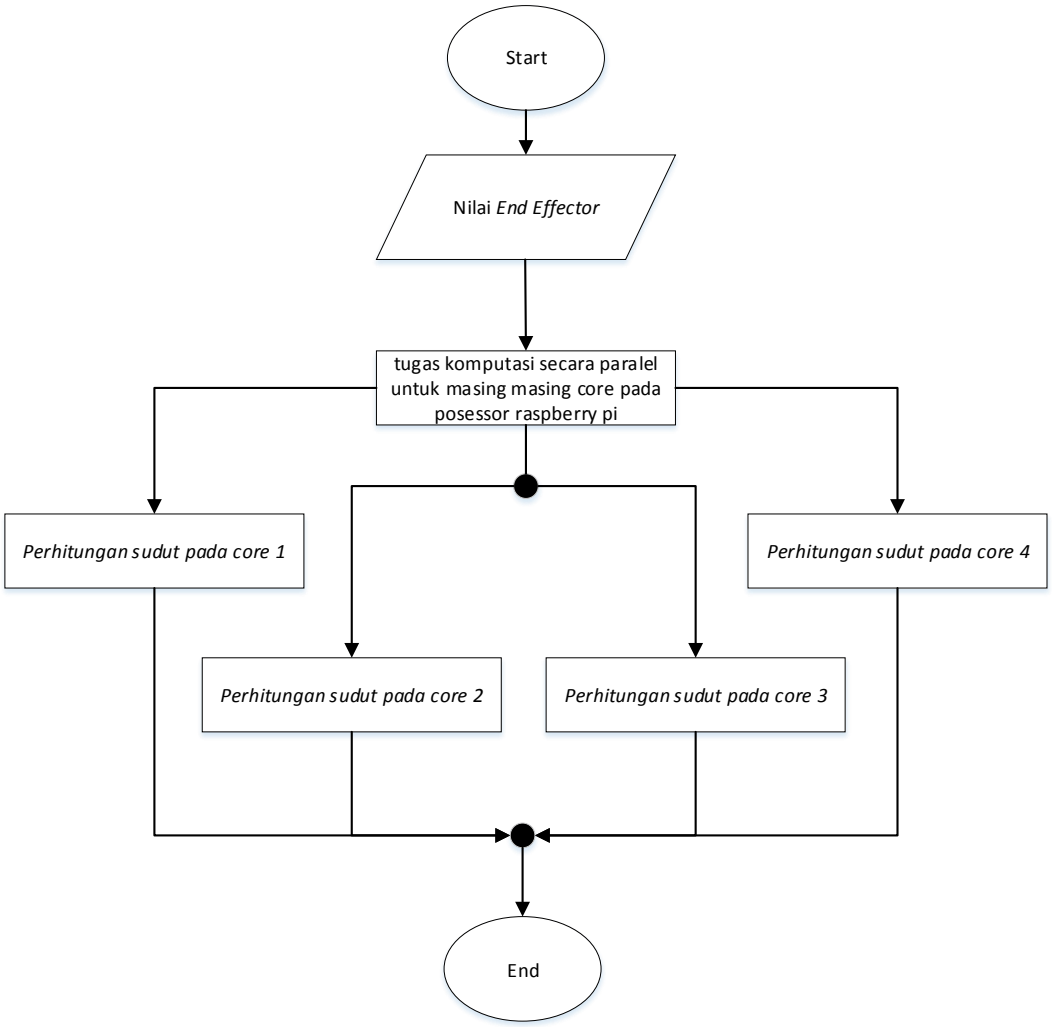
dengan pin fisik 18, GPIO 27 sama dengan pin fisik 13 dan GPIO 22 sama dengan pin fisik 15.

5.1.3 Perancangan Komputasi Paralel

Adapun cara kerja sistem secara garis besar ditunjukan pada flowchart diagram pada



Gambar 5.6 . Pada diagram tersebut, menunjukan cara kerja sistem dimana peneliti harus menentukan terlebih dahulu nilai end effector yang dituju robot lengan. Didalam proses perhitungan inverse kinematic terjadi proses pembagian tugas pada beberapa *core* prosesor. Masing masing *core* akan menghitung suatu sudut secara paralel.



Gambar 5.6 flow chart sistem

Komputasi parallel menggunakan openMP memungkinkan pembagian task di level CPU (*Central Processing Unit*). CPU yang akan digunakan tidak dapat kita tentukan, sistem akan menentukan sendiri CPU yang mampu melakukan kerja secara otomatis. OpenMP memiliki beberapa jenis *Worksharing Constructor* yang dapat digunakan. Peneliti menggunakan *Worksharing Constructor parallel section*, *Worksharing Constructor* ini dipilih karena sesuai dengan kebutuhan penelitian.

Section contructs adalah suatu metode pembagian tugas non-iteratif yang berisi satu set blok yang terdistribusi dan dieksekusi oleh sebuah thread dalam team. Setiap tugas diselesaikan oleh sebuat *thread* dalam konteks tugas implisit.

```
#pragma omp sections [clause[ [ , ] clause] ... ] new-line
    {
    [#pragma omp section new-line]
        structured-block
    [#pragma omp section new-line
        structured-block]
    ...
    }
```

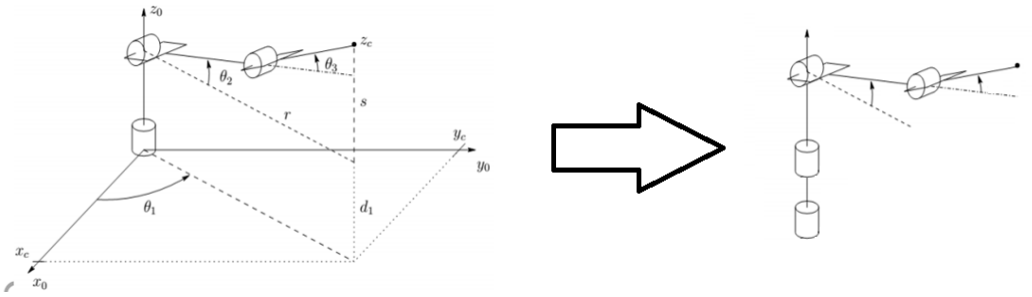
Gambar 5.7 Dokumentasi Open MP

Sumber : https://computing.llnl.gov/tutorials/openMP (OpenMP, 2015)

Masing masing tugas perhitungan nilai sudut dari *inverse kinematic* didefinisikan dalam *structured-block.* Dengan demikian perhitungan dari masing masing sudut akan dikerjakan secara terpisah dan tidak saling menunggu.

5.1.4 Perancangan Inverse Kinematic

Pada penelitian ini persamaan *Inverse Kinematic* yang digunakan menggunakan persamaan 2.1, 2.4, dan 2.5 sebagai persamaan untuk menentukan sudut.



Gambar 5.8 modifikasi persamaan inverse kinematic

Karena servo HS-322HD hanya dapat bergerak maksimal sebesar 200^o maka dilakukan modifikasi pada persamaan 2.1. Gambar 5.8 tampak terdapat perbedaan pada servo yang sejajar dengan sumbu z yang semula 1 servo menjadi 2 servo. Nilai $\boldsymbol{\theta_1}$dibagi dua ke servo 1 dan 2 sehingga maksimal putaran servo dapat mmencapai 360^o. Sehingga persamaan $\boldsymbol{\theta_1}$ menjadi :

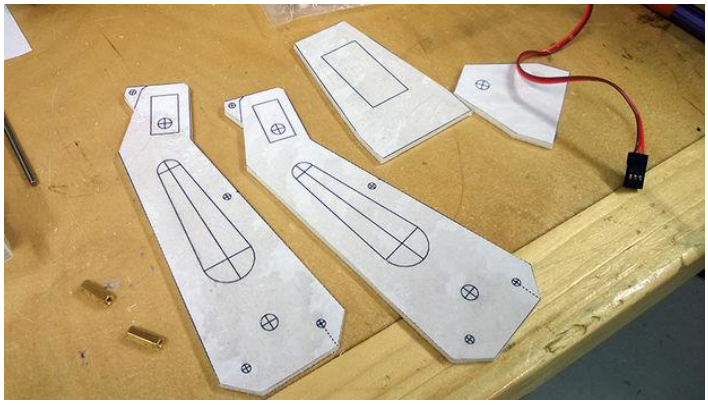
$$\frac{\boldsymbol{atan2(y,x)}}{\boldsymbol{2}} \qquad \textbf{5.1}$$

Persamaan 5.1 ddigunakan pada kedua servo yang sejajar dengan sumbu z.

5.2 Implementasi

5.2.1 Implementasi Desain Robot

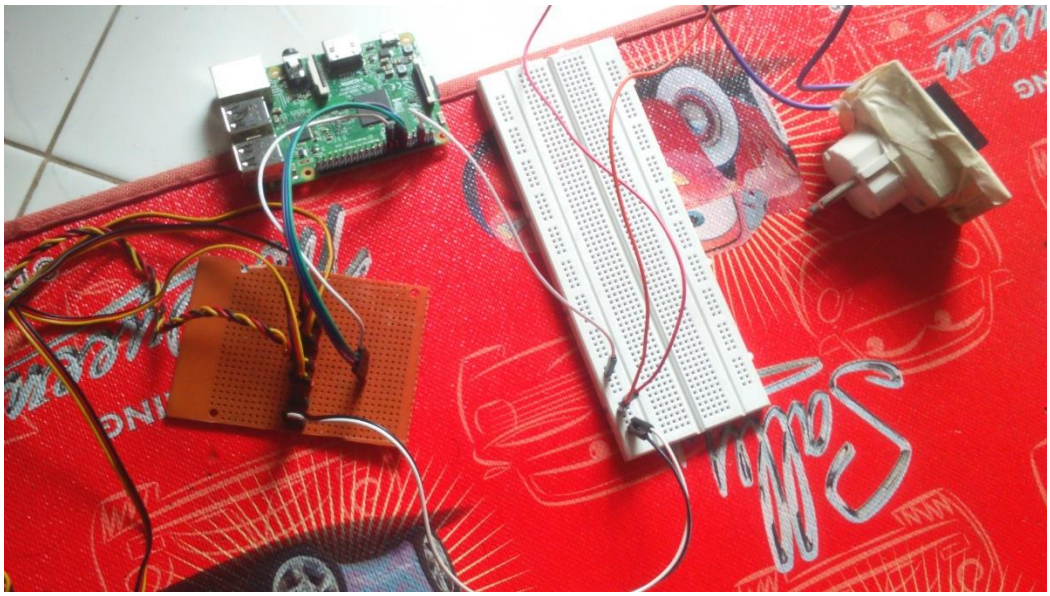
Pembuatan desain robot diawali dengan membuat gambar bagian – bagian robot secara digital. Hasil desain tersebut kemudian digunakan sebagai pola pemotongan bahan acrylic.



Gambar 5.9 hasil potongan bahan

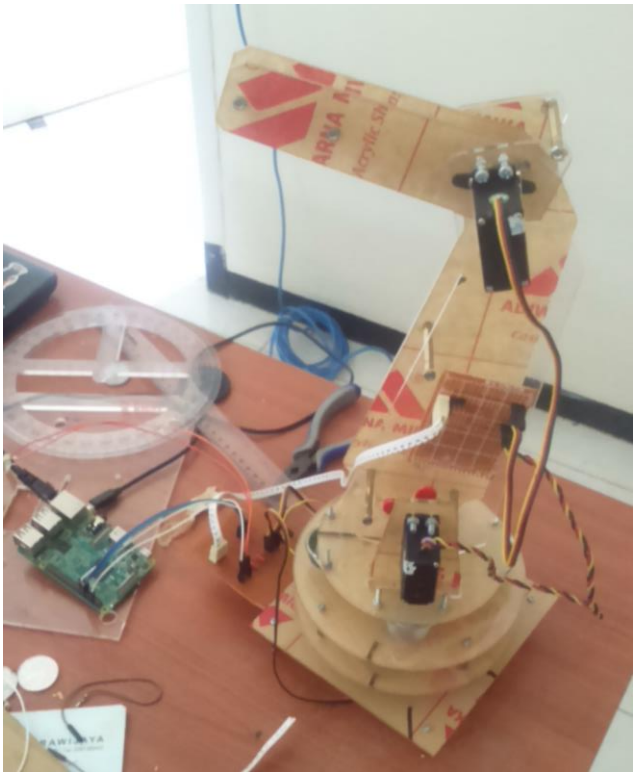
Setelah bahan dipotong seperti pada Gambar 5.9, bahan di rangkai sesuai dengan desain. Bahan dirangkai dan servo di letakkan sesuai lokasinya. Setela semua bahan selsai dirangkai, lakukan pemasangan rangkaian elektronika.

5.2.2 Implementasi rangkaian elektronika



Gambar 5.10 rangkaian elektronika

Gambar 5.10 menunjukan rangkaian elektronika yang sudah dibuat berdasarkan Tabel 5.1. Rangkaian ini kemudia di pasangkan pada body robot seperti terlihat pada Gambar 5.11.



Gambar 5.11 robot lengan

5.2.3 Implementasi Komputasi Paralel dan Inverse Kinematic

Berikut adalah potongan souce file program perhitungan inverse kinematic pada lengan robot. Secara parallel dan serial terdapat perbedaan pada penggunaan OpenMP.

Tabel 5.2 Potongan Source Code

Line	Code program
1	#pragma omp parallel sections // inisialisai slave threads
2	{
3	#pragma omp section // deklarasi thread
4	{
5	for (int i = 0; i < interfal; i++) {
6	teta1 = kali((atan2(x, y))/2, 180) / PI; // persamaan inverse kinematic
7	if (teta1 <= 0) {
8	teta1 = 360 + teta1;

```
 9              }
10            }
11
12            printf("teta 1 : %f \n ", teta1);
13            printf("Servo 0\n");
14           sudServo0 = (((teta1 / 180 * 18 / 2) + 5)); // konfersi sudut ke pwm
15            printf("sudut servo 0 : %f \n ", sudServo0);
16            softPwmWrite(0, sudServo0);
17            delay(DELAY);
18             }
19        }
20  #pragma omp section
21        {
22  .
  .   .
  .   .
  .         }
  .   }
```

Berdasarkan Tabel 5.2, Baris 1 berisi “#pragma omp parallel sections“ berfungsi untuk memberikan isialisasi bahwa section – section didalamnya berkerja secara parallel. Baris 3 “#pragma omp section” menunjukan program yang ada didalamnya merupakan sebuah section yang akan berdiri sendiri. Dalam setiap section terdapat beberapa perhitungan yang terjadi. Perhitungan pada setiap section meliputi perhitungan teta, konfersi nilai teta untuk pwm, dan perintah bergerak servo sesuai nilai pwm. Baris 5 sampai 10 merupakan rumus perhitungan teta pertama sesuai dengan persamaan (2.1) – (2.5). Baris 14 merupakan rumus konfersi nilai sudut teta 1 ke pwm servo. Baris 16 merupakan sintaks untuk menggerakan servo.

BAB 6 PENGUJIAN DAN ANALISIS

Pengujian dilakukan untuk mengetahui keberhasilan penggunaan metode sehingga didapatkan hasil yang sesuai. Pengujian dilakukan dengan beberapa macam pengambilan data. Pengujian dilakukan dengan 3 tahap yakni, pengujian waktu penyelesaian task metode parallel dibandingankan dengan metode sekuensial, pengujian akurasi titik akhir, pengujian pengaruh jumlah servo terhadap waktu penyelesaian task.

6.1 Pengujian CPU Usage

6.1.1 Tujuan Pengujian

Pengujian CPU usage bertujuan untuk membandingkan penggunaan CPU dengan metode parallel dan sekuensial. Hasil dari pengujian ini akan menampilkan seberapa besar penggunaan CPU saat program berjalan.

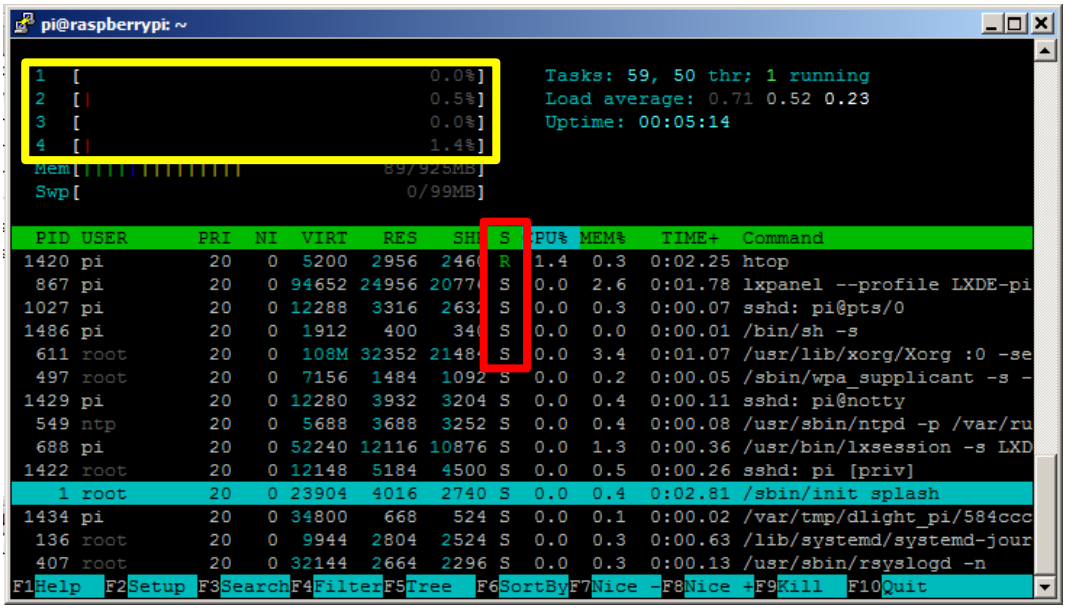
6.1.2 Proses Pengujian

Adapun proses pengujian adalah sebagai berikut :

1. Buka remote monitor putty.
2. Pada panel configurasi gunakan *connection type* SSH, masukkan Hostname IP address dari raspberry pi, Isikan “22” pada kolom port
3. Tekan tombol Open, maka akan muncul tampilan terminal dari raspberry pi.
4. Masukkan username dan password
5. Masukan command “sudo -htop”
6. Tanpa menutup window putty, buka IDE netbeans.
7. Jalankan program pergerakan robot dengan metode parallel
8. Amati perubahan CPU usage kemudian masukan kedalam tabel pengujian
9. Setelah program selesai jalankan program pergerakan robot dengan metode sekuensial
10. Amati perubahan CPU usage kemudian masukan kedalam tabel pengujian

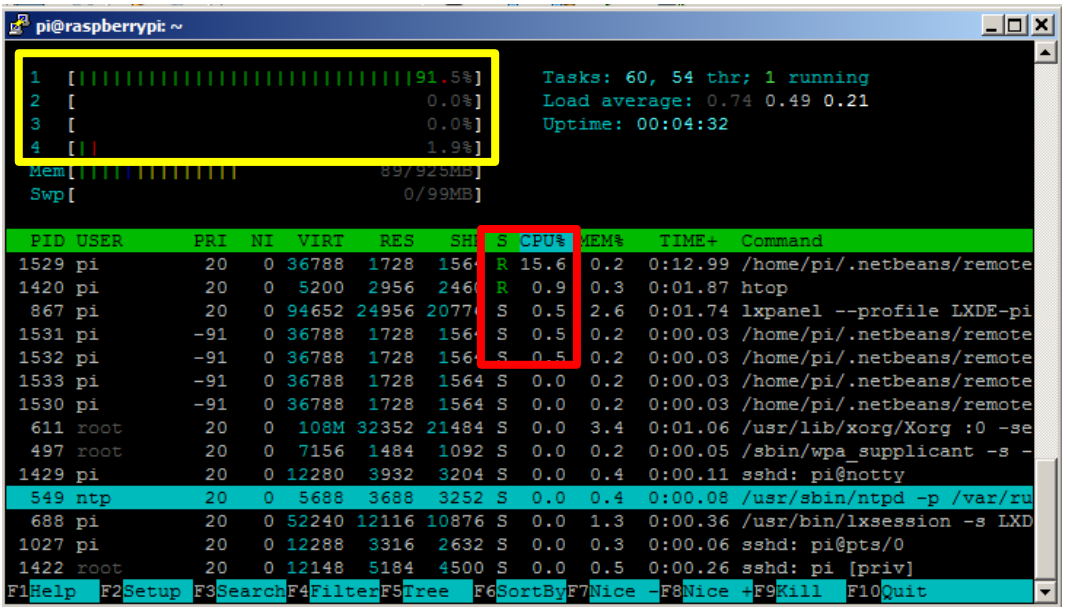
6.1.3 Hasil Pengujian

Pada tahap ini dilakukan pengujian untuk mengetahui dan membandingkan *cpu usage* saat program yang menggunakan komputasi parallel dengan program sekuensial.



Gambar 6.1 resource monitor saat program belum berjalan

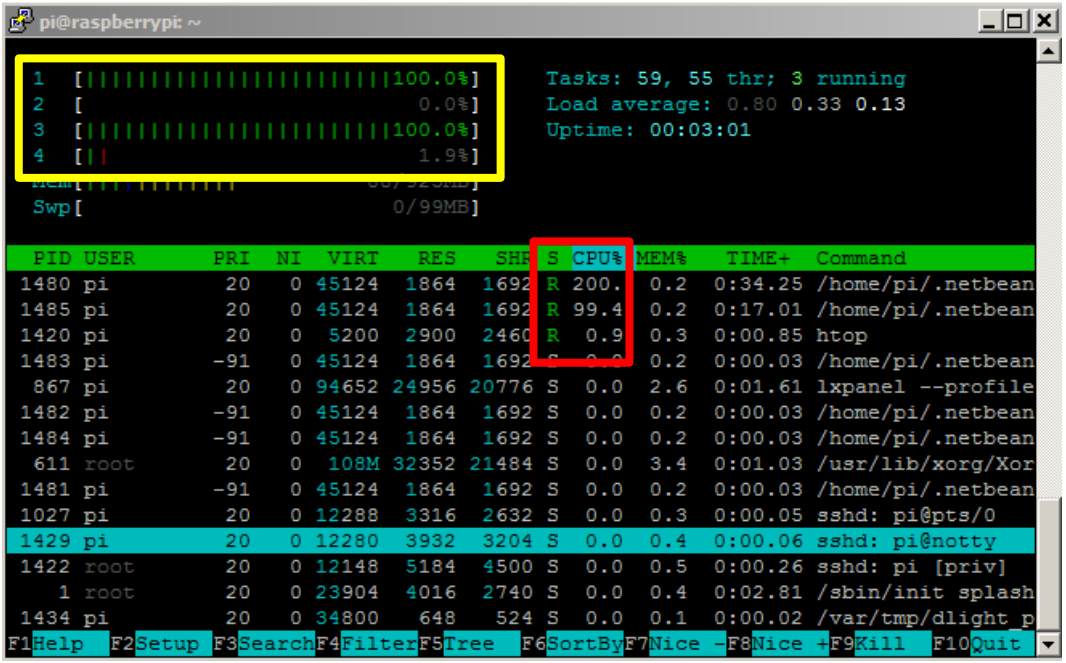
Gambar 6.1 menunjukan *resource monitor* pada raspberry pi saat program belum berjalan. Pada tanda kotak berwarna merah status running "R" hanya terdapat pada *command* htop yang merupakan menu menampilkan resource monitor itu sendiri. Pada bagian kotak berwana kuning tampak masing – masing core pada cpu tidak menjalankan program apapun selain htop yang berjalan pada core 4.



Gambar 6.2 resource monitor program sekuensial

Gambar 6.2 menunjukan resource monitor saat program sekuensial tanpa openMP dijalankan. Pada kotak merah dapat dilihat terdapat hanya satu program yang sedang running. Program tersebut menggunakan 15,6% cpu untuk

menjalankan programnya. Pada kotak berwarna kuning core yang menjalankan program tersebut adalah core 1 saja. Karena program pergerakan lengan robot yang digunakan pada pengujian ini tidak menggunakan komputasi parallel.



Gambar 6.3 resource monitor program dengan parallel

Gambar 6.3 menunjukan resource monitor saat program dengan openMP dijalankan. Pada kotak merah tampak ada 2 program yang berada pada kondisi running (“R”). Program tersebut masing masing menggunakan 200% dan 99% cpu untuk menjalankan programnya. Pada kotak kuning tampak core 1 dan core 3 menjalankan program secara bersamaan. hal ini terjadi karena openMP mengatur program agar setiap section dijalankan secara bersamaan pada core yang tersedia. Pengujian ini menunjukan program pergerakan lengan robot dikerjakan secara parallel.

Tabel 6.1 pengujian cpu usage 1

CPU usage program sekuensial				
Pengujian	Core 1	Core 2	Core 3	Core 4
1	0	51,7	0,5	0,9
2	0,5	30,5	19,4	0,5
3	43	0,5	0	0,9
4	61,4	0	0	1,2
5	48,8	0,5	0	0,9
Rata -rata	30,74	16,64	3,98	0,9

Tabel 6.2pengujian cpu usage 2

CPU usage program parallel				
pengujian	Core 1	Core 2	Core 3	Core 4
1	0,5	0,5	16,2	18,4
2	0	33,3	53,3	1,4
3	0,5	49,5	14,6	0,5
4	0	52,6	17,8	0,5
5	51,7	17,6	0	0
Rata - rata	10,54	30,7	20,38	4,16

Berdasarkan data dari Tabel 6.1 program sekuensial membebankan prosesnya pada core 1 (30,74 %), sedangkan berdasarkan Tabel 6.2, pada program parallel beban prosesnya cukup terbagi rata meskipun terlihat lebih membebankan pada core 2 (30,7%) dan 3 (20,38 %). Core yang dipilih untuk dibebankan tugas komputasi tidak dapat ditentukan. Dengan metode OpenMP *parallel section* CPU yang dibebankan tugas komputasi di generate secara otomatis oleh OpenMP.

6.2 Pengujian waktu

6.2.1 Tujuan Pengujian

Pengujian waktu bertujuan untuk mengetahui perbedaan waktu penyelesaian tugas dari metode parallel dengan meotde sekuensial. Dengan tugas berupa robot dapat mencapai 5 titik yang sama, berapa lama waktu yang dibutuhkan oleh masing - masing metode untuk menyelesaikanya.

6.2.2 Proses Pengujian

Adapun proses pengujian adalah sebagai berikut :

1. Buka IDE netbeans.
2. Jalankan program pergerakan robot dengan metode parallel
3. Amati total time yang muncul di output windows pada netbeans kemudian masukan kedalam tabel pengujian.
4. Setelah program selesai jalankan program pergerakan robot dengan metode sekuensial
5. Amati total time yang muncul di output windows pada netbeans kemudian masukan kedalam tabel pengujian.

6.2.3 Hasil Pengujian

Pengujian ini dilakukan dengan melihat *run time* dari netbeans setelah task dilakukan. Task terdiri dari 5 titik *end efector* yang kemudian dituju oleh robot

lengan. Pengujian dilakukan sebanyak 5 kali. Waktu yang tertulis bukan waktu kerja mesin secara *real time* penulis berpedoman pada waktu yang ditampilkan dari IDE netbeans yang merupakan gabungan dari waktu *compile,* waktu membentuk koneksi *serial monitor* dan waktu kerja yang dilakukan robot.

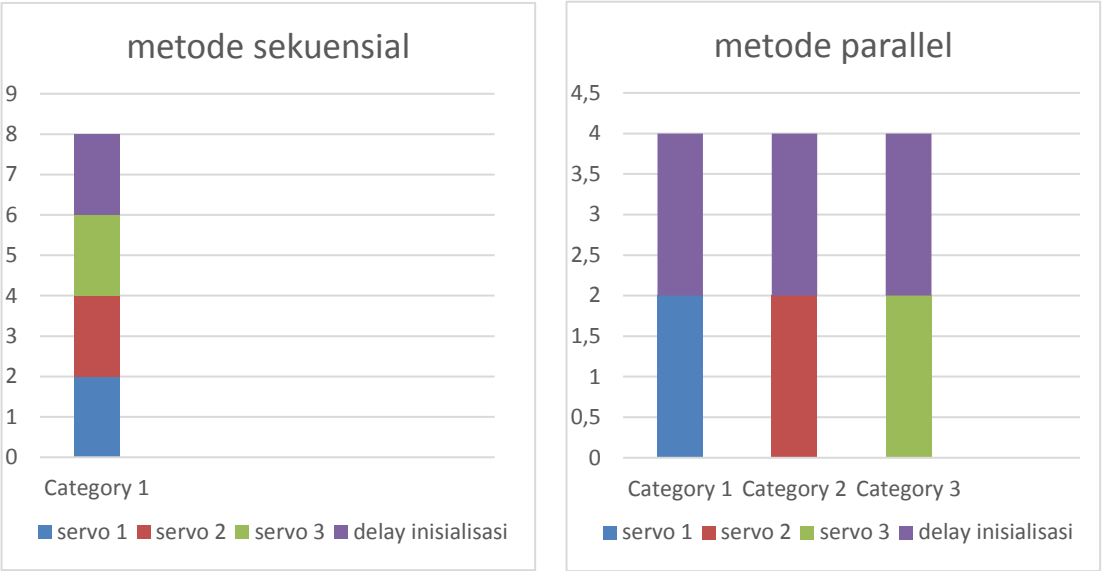
Tabel 6.3 pengujian waktu dengan 4 servo

	parallel	sekuensial	Selisih
Pengujian 1	35 detik	55 detik	20 detik
Pengujian 2	35 detik	55 detik	20 detik
Pengujian 3	35 detik	55 detik	20 detik
Pengujian 4	35 detik	55 detik	20 detik
Pengujian 5	35 detik	55 detik	20 detik
Rata – rata	35 detik	55 detik	20 detik

Tabel 6.4 tabel pengujian waktu dengan 3 servo

	parallel	sekuensial	Selisih
Pengujian 1	35 detik	45 detik	10 detik
Pengujian 2	35 detik	45 detik	10 detik
Pengujian 3	35 detik	45 detik	10 detik
Pengujian 4	35 detik	45 detik	10 detik
Pengujian 5	35 detik	45 detik	10 detik
Rata – rata	35 detik	45 detik	10 detik

Berdasarkan pengujian pada Tabel 6.3 dan Tabel 6.4 terlihat bahwa terjadi perbedaan waktu penyelesaian pada kedua metode. Metode parallel dapat menyelesaikan task dengan lebih cepat. Selisih waktu yang terjadi adalah 20 detik (36,4% dari waktu sequensial) dan 10 detik (22,2% dari waktu sekuensial). Hal ini terjadi karena setiap servo membutuhkan delay untuk bergerak, delay ini akan sangat berpengaruh pada waktu penyelesaian task. Pada metode parallel jumlah servo tidak bedampak memperlambat perhitungan waktu kerja program. Semakin banyak servo yang digunakan akan berdampak memperlambat perhitungan kerja pada metode sekuensial. Perbedaan yang terjadi dijelaskan pada Gambar 6.4.



Gambar 6.4 perbandingan waktu kerja robot

Gambar 6.4 menunjukan perbedaan proses penerjaan tugas dari masing2 metode. Pada proses sekuensial servo 2 baru akan dikerjakan bila servo 1 selesai bererak dan servo 3 hanya akan bererak setelah servo 2 selesai bererak. Sedangkan pada program parallel servo 1 dan 2 dapat bergerak bersamaan.

6.3 Pengujian akurasi

6.3.1 Tujuan Pengujian

Pengujian akurasi bertujuan untuk mengetahui seberapa akurat gerakan robot sesungguhnya. Pengujian ini dilakukan dengan mengukur sudut servo dengan busur derajat dan lokasi titik akhir robot dengan penggaris mistar.

6.3.2 Proses Pengujian

Adapun proses pengujian adalah sebagai berikut :

1. Buka IDE netbeans
2. Buat program agar hanya bergerak pada satu titik akhir.
3. Setelah robot berada pada posisi ikatkan sebuah tali dengan pemberat pada ujung robot
4. Setelah tali dalam posisi setimbang hitung panjang sumbu x, y dan z, dengan mengacu pada lokasi tali dan pemberat. Masukkan nilai x, y dan z ke dalam tabel pengujian.
5. Beri tanda pada posisi 0^o servo. Ukur sudut yang dihasilkan masing masing servo dengan busur derajat. Masukkan nilai sudut servo 1, 2, dan 3 ke dalam tabel pengujian

6.3.3 Hasil Pengujian

Pada pengujian ini dilakukan perhitungan lokasi *end efector* dengan membandingkan gerak robot berdasarkan *inverse kinematic* yang telah dihitung dengan keadaan sesungguhnya.

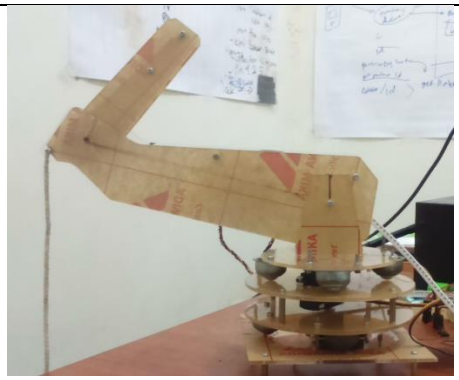
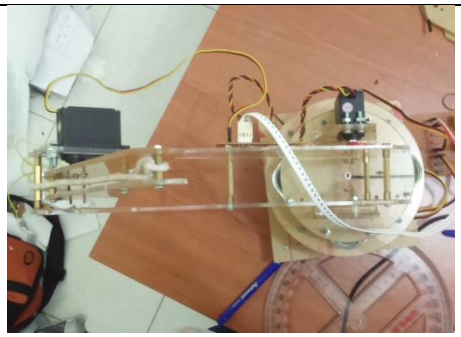
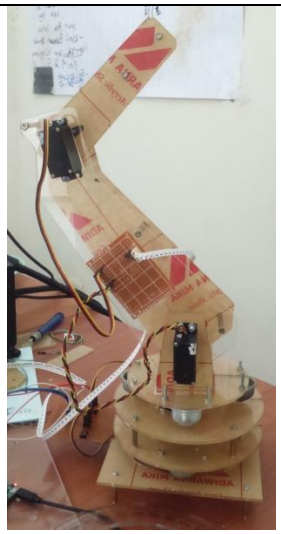
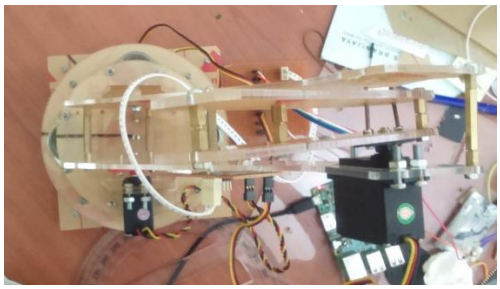
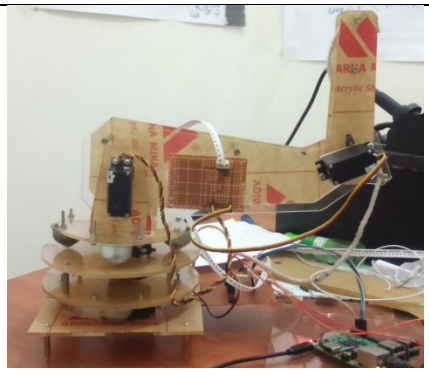
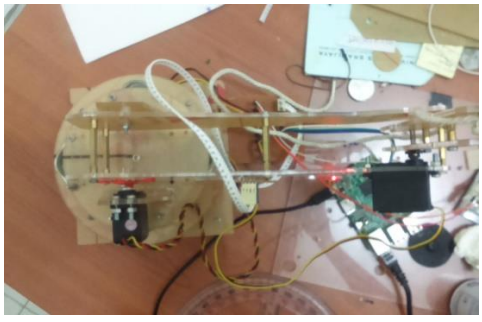
Tabel 6.5 pengujian akurasi titik akhir

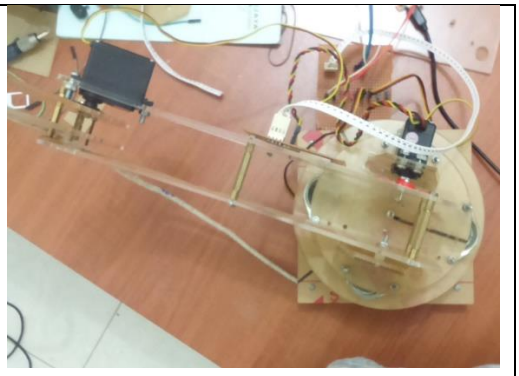
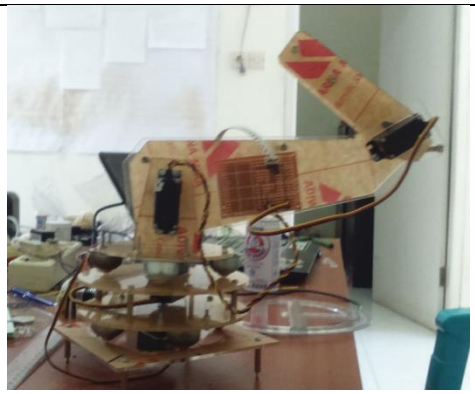
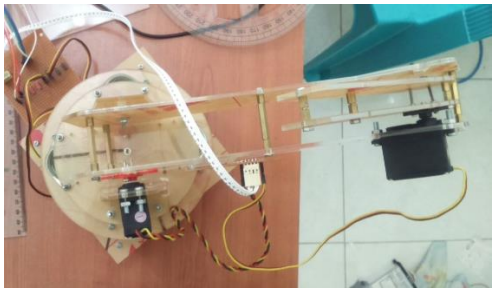
No	Titik tujuan (x,y,z)	hasil pengukuran paralle	**Error x**	**Error y**	**Error z**	hasil pengukuran sekuensial	**Error x**	**Error y**	**Error z**
1	0, -15, 30	0,12.5,31	0	2.5	1	0,12.5,31	0	2.5	1
2	0, 2, 40	0, 1 ,39.7	0	1	0.3	0, 1 ,40	0	1	0
3	0, 21, 27	0, 21 , 27	0	0	0	0, 21 , 27	0	0	0
4	-21, 10, 27	-23, 7, 29	2	3	2	-23, 8, 29	2	3	2
5	-5, 8, 26	-5, 9, 28	0	1	2	-5, 9, 28	0	1	2
Rata – rata error			0.4	1.5	1.06		0.4	1.5	1

Tabel 6.6 pengujian akurasi sudut

no	Sudut (1,2,3)	Hasil pengukuran	Error 1	Error 2	Error 3	hasil pengukuran sekuensial	Error 1	Error 2	Error 3
1	180,12,105	180,15,110	0	3	5	180,15,110	0	3	5
2	0,57,85	0, 59, 88	0	2	3	0, 59, 88	0	2	3
3	0, 0, 90	0, 0, 90	0	0	0	0, 0, 90	0	0	0
4	295, 0, 78.5	290, 0, 74	5	0	4.5	290, 0, 74	5	0	4.5
5	328, 15.8, 138	328, 17, 130	0	1.2	8	328, 17, 130	0	1.2	8
Rata – rata error			1	1.24	4.1		1	1.24	4.1

Tabel 6.7 pengujian akurasi 3

No	Tampak samping	Tampak atas
1		
2		
3		

4		
5		

Berdasarkan Tabel 6.5, Tabel 6.6 dan Tabel 6.7 menunjukan *inverse kinematic* dapat diimplementasikan pada robot manipulator dengan menggunakan raspberry pi. Berdasarkan pengujian, terdapat rata – rata error sebesar 0.4cm dari sumbu x, 1.5 cm dari sumbu y dan 1.06 cm dari sumbu z. Selain itu error juga terjadi pada sudut servo, dimana rata – rata error pada sudut 1 sebesar 1^{o}, pada sudut 2 sebesar 1.24^{o}, dan pada sudut 3 sebesar 4.1^{o}.

Error yang terjadi dapat disebabkan beberapa faktor antara lain, kalibarasi mekanika yang kurang sempurna, faktor internal servo, elektronika yang kurang stabil, dan pengukuran yang kurang presisi.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari tahapan perancangan, implementasi, pengujian serta analisis hasil pengujian yang telah dilakukan, penulis dapat menarik kesimpulan bahwa :

1. Metode *Inverse Kinematic* dapat di implementasikan pada robot manipulator dengan menggukana kontroler raspberry pi. Sudut dudut hasil perhitungan *inverse kinematic* dapat di eksekusi dengan menggunakan fitur *softpwm* yang disediakan wiring pi pada raspberry. Melalui *softpwm* servo hi-tech HS-322HD dapat digunakan sebagai penggerak robot manipulator. Pergerakan robot bila dibandingkan, metode sekuensial melakukan gerakan yang lebih halus. Akurasi dari kedua metode tidak jauh berbeda, keduanya sama sama memiliki nois dan error yang sama yaitu sebesar 0.4cm dari sumbu x, 1.5 cm dari sumbu y dan 1.06 cm dari sumbu z. Selain itu error juga terjadi pada sudut servo, dimana rata – rata error pada sudut 1 sebesar 1^o, pada sudut 2 sebesar 1.24^o, dan pada sudut 3 sebesar 4.1^o. Bila dibandingkan dari segi kecepatan, metode parallel menjalankan seluruh tugas dengan lebih cepat 20 detik (36,4% dari waktu sequensial) untuk 4 servo dan 10 detik (22,2% dari waktu sekuensial) untuk 3 servo.
2. Raspberry pi mampu membagi tugas komputasi dengan memanfaatkan 4 core yang ada didalamnya. Multycore programing dengan openMP terbukti mampu memanfaatkan hampir seluruh resource yang di sediakan CPU milik raspberry pi. Program sekuensial membebankan prosesnya pada core 1 (30,74 %). Pada program parallel beban prosesnya cukup terbagi rata meskipun terlihat lebih membebankan pada core 2 (30,7%) dan 3 (20,38 %).
3. Implementasi komputasi parallel pada raspberry pi dapat dilakukan dengan menggunakan liblary yang disebut openMP. Nilai sudut sudut hasil perhitungan Inverse Kinematic di hitung secara terpisah terbukti dengan nilai sudut akan tampil setelah hasil perhitungan selesai. Rumus yang lebih sederhana dan menghasilkan keluaran yang lebih cepat.

7.2 Saran

Berdasarkan kesimpulan yang telah didapatkan, terdapat beberapa saran untuk penelitian selanjutnya sebagai langkah pengembangan sistem ini, yaitu :

1. Karena beberapa faktor, gerakan servo masih memiliki error sehingga disarankan agar memanfaatkan metode PID () untuk menambah akurasi gerak servo.
2. Penggunaan openMP tergolong mudah sehingga disarankan untuk mengimplementasikan openMP untuk komputasi parallel pada sistem yang kompleks dan kontinue untuk menguji lebih lanjut efektifitas openMP
3. Untuk penelitian lebih lanjut disarankan untuk menggukan inverse kinemapit secara parralel pada servo yang lebih banyak.
4. Disarankan menggunakan metode pengukuran sudut dan titik tujuan dengan menggunakan sensor agar pengukuran dapat dilakukan dengan lebih akurat.

DAFTAR PUSTAKA

College, D., 2009. *What is OpenMP.* [Online] Available at: https://www.dartmouth.edu [Accessed 15 januari 2018].

Desiani, 2015. *APLIKASI SENSOR PROXIMITY PADA LENGAN ROBOT SEBAGAI PENYORTIR KOTAK BERDASARKAN UKURAN BERBASIS ARDUINO UNO,* Palembang: Politeknik Negeri Sriwijaya.

Dingju Zhu, J. F., 2008. *Application of Parallel Computing in Digital City,* Dalian, China: IEEE.

Mark W. Spong, S. H. a. M. V., 2005. *Robot Modeling and Control.* 1st ed. New York / Chichester / Weinheim / Brisbane / Singapore / Toronto: JOHN WILEY & SONS, INC..

Maulana, R., 2015. *05. Forward Kinematics [Power Point slides]* [Interview] (2 Desember 2015).

OpenMP, 2015. *OpenMP Application Programming Interface.* 4.5 November 2015 ed. s.l.:OpenMP Architecture Review Board.

Peter Arbenz, W. P., 2011. *C++ Examples of Parallel Programming with OpenMP.* [Online] Available at: https://people.sc.fsu.edu/~jburkardt/cpp_src/openmp/openmp.html

RobotFreak, 2017. *Let's Make Robots.* [Online] Available at: http://www.robotshop.com/letsmakerobots/ax-12-robot-arm

RobotWorx, 2017. [Online] Available at: https://www.robots.com/faq/show/what-is-a-robot-manipulator

Rus, D., 2011. *Robotics systems and science, Lecture 14: Forward and Inverse* [Interview] (Spring 2011).

Setiawan, E., 2014. *Robotika [Power Point slides]* [Interview] 2014.

Setiawan, S., Rahmadya, B., F. & D., 2015. PENERAPAN INVERS KINEMATIKA UNTUK PERGERAKAN KAKI ROBOT. *JURNAL FTUMJ.*

Zarkasih, M., Wibisono, W. & Arunanto, F., 2013. Implementasi Komputasi Paralel Untuk Enkripsi. *JURNAL TEKNIK POMITS,* pp. 1-2.

LAMPIRAN A SOURCE CODE PROGRAM PARALLE

Nomor baris	Source Code
1	#include <wiringPi.h>
2	#include <softPwm.h>
3	#include <omp.h>
4	#include <stdio.h>
5	#include <stdlib.h>
6	#include <math.h>
7	
8	
9	#define PI 3.14159265
10	#define DELAY 1000
11	
12	const float L1 = 13; // set untuk panjang link
13	const float L2 = 21;
14	const float L3 = 12;
15	const int interfal = 100;
16	float teta1, teta2, teta3;
17	float s3, c3;
18	float a, b, temp;
19	int num;
20	float sudServo0, sudServo2, sudServo3, sudServo1 = 0;
21	
22	
23	void hitung(float x, float y, float z);
24	
25	int main(int argc, char *argv[]) {
26	if (wiringPiSetup() == -1) {
27	printf("error");
28	exit(1);

```
29      }
30
31      softPwmCreate(0, 14, 100);
32      softPwmCreate(2, 15, 100);
33      softPwmCreate(1, 1, 100);
34      softPwmCreate(3, 1, 100);
35
36      delay(5000);
37
38      hitung(0, 20, 25);
39      delay(1000);
40      hitung(0, 20, 35);
41      delay(1000);
42      hitung(20, 10, 35);
43      delay(1000);
44      hitung(20, 0, 35);
45      delay(1000);
46      hitung(20, 0, 25);
47      delay(1000);
48
49      return 0;
50  }
51
52  float pangkat(float a, float b) {
53      float hasil = 1;
54      for (char i = 0; i < b; i++) {
55          hasil = hasil * a;
56      }
57      return hasil;
58  }
59
60  float akar(float a) {
```

```
61      return sqrt(a);
62  }
63
64  float kali(float a, float b) {
65      float hasil = a*b;
66      return hasil;
67  }
68
69  void hitung(float x, float y, float z) {
70      //    {
71      c3 = ((pangkat(x, 2.0) + pangkat(y, 2.0) + pangkat(z - L1, 2.0) -
72  pangkat(L2, 2.0) - pangkat(L3, 2.0))) / kali(kali(2, L2), L3);
73      s3 = (akar(1 - pangkat(c3, 2.0)));
74      printf("nilai s3 : %f \n ", s3);
75
76  #pragma omp parallel sections
77      {
78  #pragma omp section
79          //teta 1
80          {
81              for (int i = 0; i < interfal; i++) {
82                  teta1 = kali(atan2(x, y), 180) / PI;
83                  if (teta1 < 0) {
84                      teta1 = 360 + teta1;
85                  }
86              }
87
88              printf("teta 1 : %f \n ", teta1);
89              printf("Servo 0\n");
90              sudServo0 = (((teta1 / 180 * 18 / 2) + 6));
91              printf("sudut servo 0 : %f \n ", sudServo0);
92              softPwmWrite(1, sudServo0);
```

```
93          delay(DELAY);
94      }
95  #pragma omp section
96      {
97          for (int i = 0; i < interfal; i++) {
98              teta1 = kali(atan2(x, y), 180) / PI;
99              if (teta1 < 0) {
100                 teta1 = 360 + teta1;
101             }
102         }
103         sudServo3 = ((teta1 / 180 * 18 / 2) + 5);
104         softPwmWrite(3, sudServo3);
105         delay(DELAY);
106     }
107
108 #pragma omp section
109     {
110         for (int i = 0; i < interfal; i++) {
111             a = kali(atan2((z - L1), akar((pangkat(x, 2.0) + pangkat(y,
    2.0)))), 180) / 3.14;
112
113             b = kali(atan2(kali(L3, s3), (L2 + kali(L3, c3))), 180) / 3.14;
114             teta2 = a - b;
115             teta2 = teta2 + 45;
116             if (teta2 < 0) {
117                 teta2 = 360 + teta2;
118             }
119             if (teta2 > 360) {
120                 teta2 = teta2 - 360;
121             }
122         }
123         teta2=-teta2;
124         printf("teta 2 : %f\n ", teta2);
```

```
125          printf("Servo 1\n");
126          sudServo1 = (teta2 / 180 * 18) + 23;
127          printf("sudut servo 1 : %f \n ", sudServo1);
128          softPwmWrite(2, sudServo1);
129          delay(DELAY);
130       }
131
132    #pragma omp section
133       {
134          for (int i = 0; i < interfal; i++) {
135             teta3 = (kali(atan2(s3, c3), 180) / 3.14);
136             if (teta3 < 0) {
137                teta3 = 360 + teta3;
138             }
139          }
140          teta3 = -teta3;
141          printf("teta 3 : %f\n ", teta3);
142          printf("Servo 3\n");
143          sudServo2 = (teta3 / 180 * 18) + 23;
144          printf("sudut servo 2 : %f  ", sudServo2);
145          softPwmWrite(0, sudServo2);
146          delay(DELAY);
147       }
148    }
149 }
150
```

LAMPIRAN B SOURCE CODE PROGRAM SEKUENSIAL

Nomor baris	Source code
1	#include <wiringPi.h>
2	#include <softPwm.h>
3	#include <stdio.h>
4	#include <stdlib.h>
5	#include <math.h>
6	
7	
8	#define PI 3.14159265
9	#define DELAY 1000
10	
11	const float L1 = 15; // set untuk panjang link
12	const float L2 = 21;
13	const float L3 = 12;
14	const int interfal = 1000000;
15	float teta1, teta2, teta3;
16	float s3, c3;
17	float a, b, temp;
18	int num;
19	float sudServo0, sudServo2, sudServo3, sudServo1 = 0;
20	
21	
22	void hitung(float x, float y, float z);
23	
24	int main(int argc, char *argv[]) {
25	if (wiringPiSetup() == -1) {
26	printf("error");
27	exit(1);
28	}

```
29
30      softPwmCreate(0, 14, 100);
31      softPwmCreate(2, 14, 100);
32      softPwmCreate(1, 1, 100);
33      softPwmCreate(3, 1, 100);
34
35      delay(5000);
36
37      hitung(0, 20, 25);
38      delay(1000);
39      hitung(0, 20, 35);
40      delay(1000);
41      hitung(20, 10, 35);
42      delay(1000);
43      hitung(20, 0, 35);
44      delay(1000);
45      hitung(20, 0, 25);
46      delay(1000);
47
48      return 0;
49  }
50  float pangkat(float a, float b) {
51      float hasil = 1;
52      for (char i = 0; i < b; i++) {
53          hasil = hasil * a;
54      }
55      return hasil;
56  }
57  float akar(float a) {
58      return sqrt(a);
59  }
60
```

```
float kali(float a, float b) {
   float hasil = a*b;
   return hasil;
}

void hitung(float x, float y, float z) {



   c3 = ((pangkat(x, 2.0) + pangkat(y, 2.0) + pangkat(z - L1, 2.0) - pangkat(L2, 2.0) - pangkat(L3, 2.0))) / kali(kali(2, L2), L3);
   s3 = akar(1 - pangkat(c3, 2.0));



   for (int i = 0; i < interfal; i++) {
      teta1 = kali(atan2(x, y), 180) / PI;
      if (teta1 < 0) {
         teta1 = 360 + teta1;
      }
   }

   printf("teta 1 : %f \n ", teta1);
   printf("Servo 0\n");
   sudServo0 = (((teta1 / 180 * 18 / 2) + 5));
   printf("sudut servo 0 : %f \n ", sudServo0);
   softPwmWrite(1, sudServo0);
   delay(DELAY);
   for (int i = 0; i < interfal; i++) {
      teta1 = kali(atan2(x, y), 180) / PI;
      if (teta1 < 0) {
         teta1 = 360 + teta1;
      }
   }
```

```
93    sudServo3 = ((teta1 / 180 * 18 / 2) + 5);
94    softPwmWrite(3, sudServo3);
95    delay(DELAY);
96
97    for (int i = 0; i < interfal; i++) {
98       a = kali(atan2((z - L1), akar((pangkat(x, 2.0) + pangkat(y, 2.0)))),
180) / 3.14;
99
100      b = kali(atan2(kali(L3, s3), (L2 + kali(L3, c3))), 180) / 3.14;
101      teta2 = a - b;
102      teta2 = teta2 + 45;
103      if (teta2 < 0) {
104         teta2 = 360 + teta2;
105      }
106      if (teta2 > 360) {
107         teta2 = teta2 - 360;
108      }
109   }
110   printf("teta 2 : %f\n ", teta2);
111   printf("Servo 1\n");
112   sudServo1 = (teta2 / 180 * 18) + 6;
113   printf("sudut servo 1 : %f \n ", sudServo1);
114   softPwmWrite(2, sudServo1);
115   delay(DELAY);
116
117   for (int i = 0; i < interfal; i++) {
118      teta3 = kali(atan2(s3, c3), 180) / 3.14;
119      if (teta3 < 0) {
120         teta3 = 360 + teta3;
121      }
122   }
123   teta3=-teta3;
124   printf("teta 3 : %f\n ", teta3);
```

```
125     printf("Servo 3\n");
126     sudServo2 = (teta3 / 180 * 18) + 23;
127     printf("sudut servo 2 : %f  ", sudServo2);
128     softPwmWrite(0, sudServo2);
129     delay(DELAY);
    }
```